Lecture Notes in Computer Science 2073
Edited by G. Goos, J. Hartmanis and J. van Leeuwen

Springer
Berlin
Heidelberg
New York
Barcelona
Hong Kong
London
Milan
Paris
Singapore
Tokyo

Vassil N. Alexandrov Jack J. Dongarra
Benjoe A. Juliano René S. Renner
C. J. Kenneth Tan (Eds.)

Computational Science – ICCS 2001

International Conference
San Francisco, CA, USA, May 28-30, 2001
Proceedings, Part I

Springer

Volume Editors

Vassil N. Alexandrov
University of Reading
School of Computer Science, Cybernetics and Electronic Engineering
Whiteknights, P.O. Box 225, Reading RG6 6AY, UK
E-mail: V.N.Alexandrov@rdg.ac.uk

Jack J. Dongarra
University of Tennessee
Innovative Computing Lab, Computer Science Department
1122 Volunteer Blvd, Knoxville, TN 37996-3450, USA
E-mail: dongarra@cs.utk.edu

Benjoe A. Juliano
René S. Renner
Computer Science Department, California State University
Chico, CA 95929-0410, USA
E-mail:{Juliano/renner}@ecst.csuchico.edu

C. J. Kenneth Tan
The Queen's University of Belfast
School of Computer Science
Belfast BT7 1NN, Northern Ireland, UK
E-mail: cjtan@acm.org

Cataloging-in-Publication Data applied for

Die Deutsche Bibliothek - CIP-Einheitsaufnahme

Computational science : international conference ; proceedings / ICCS 2001, San Francisco, CA, USA, May 28 - 30, 2001. Vassil N. Alexandrov ... (ed.). - Berlin ; Heidelberg ; New York ; Barcelona ; Hong Kong ; London ; Milan ; Paris ; Singapore ; Tokyo : Springer
Pt. 1 . - (2001)
(Lecture notes in computer science ; Vol. 2073)
ISBN 3-540-42232-3

CR Subject Classification (1998):D, F, G. H. I, J

ISSN 0302-9743
ISBN 3-540-42232-3 Springer-Verlag Berlin Heidelberg New York

This work is subject to copyright. All rights are reserved, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, re-use of illustrations, recitation, broadcasting, reproduction on microfilms or in any other way, and storage in data banks. Duplication of this publication or parts thereof is permitted only under the provisions of the German Copyright Law of September 9, 1965, in its current version, and permission for use must always be obtained from Springer-Verlag. Violations are liable for prosecution under the German Copyright Law.

Springer-Verlag Berlin Heidelberg New York
a member of BertelsmannSpringer Science+Business Media GmbH

http://www.springer.de

© Springer-Verlag Berlin Heidelberg 2001
Printed in Germany

Typesetting: Camera-ready by author
Printed on acid-free paper SPIN 10781763 06/3142 5 4 3 2 1 0

Preface

Computational Science is becoming a vital part of many scientific investigations, affecting researchers and practitioners in areas ranging from aerospace and automotive, to chemistry, electronics, geosciences, to mathematics, and physics. Due to the sheer size of many challenges in computational science, the use of high performance computing, parallel processing, and sophisticated algorithms, is inevitable.

These two volumes (Lecture Notes in Computer Science volumes 2073 and 2074) contain the proceedings of the 2001 International Conference on Computational Science (ICCS 2001), held in San Francisco, California, USA, May 27–31, 2001. These two volumes consist of more than 230 contributed and invited papers presented at the meeting. The papers presented here reflect the aims of the program committee to bring together researchers and scientists from mathematics and computer science as basic computing disciplines, researchers from various application areas who are pioneering advanced applications of computational methods to sciences such as physics, chemistry, life sciences, and engineering, arts and humanitarian fields, along with software developers and vendors, to discuss problems and solutions in the area, to identify new issues, and to shape future directions for research, as well as to help industrial users apply various advanced computational techniques. The aim was also to outline a variety of large-scale problems requiring interdisciplinary approaches and vast computational efforts, and to promote interdisciplinary collaboration.

The conference was organized by the Department of Computer Science at California State University at Chico, the School of Computer Science at The Queen's University of Belfast, the High Performance Computing and Communication group from the Department of Computer Science, The University of Reading, and the Innovative Computing Laboratory at the University of Tennessee. This is the first such meeting and we expect a series of annual conferences in Computational Science.

The conference included 4 tutorials, 12 invited talks, and over 230 contributed oral presentations. The 4 tutorials were "Cluster Computing" given by Stephen L. Scott, "Linear Algebra with Recursive Algorithms (LAWRA)" given by Jerzy Waśniewski, "Monte Carlo Numerical Methods" given by Vassil Alexandrov and Kenneth Tan, and "Problem Solving Environments" given by David Walker. The constitution of the interesting program was due to the invaluable suggestions of the members of the ICCS 2001 Program Committee. Each contributed paper was refereed by at least two referees. We are deeply indebted to the members of the program committee and all those in the community who helped us form a successful program. Thanks also to Charmaine Birchmore, James Pascoe, Robin Wolff, and Oliver Otto whose help was invaluable.

We would like to thank our sponsors and partner organizations, for their support, which went well beyond our expectations. The conference was sponsored by Sun Microsystems (USA), IBM (UK), FECIT (Fujitsu European Center for Information Technology) Ltd. (UK), American Mathematical Society (USA), Pacific Institute for the Mathematical Sciences (Canada), Springer-Verlag GmbH,

California State University at Chico (USA), The Queen's University of Belfast (UK), and The University of Reading (UK).

ICCS 2001 would not have been possible without the enthusiastic support of our sponsors and our colleagues from Oak Ridge National Laboratory, University of Tennessee and California State University at Chico. Warm thanks to James Pascoe, Robin Wolff, Oliver Otto, and Nia Alexandrov for their invaluable work in editing the proceedings; to Charmaine Birchmore for dealing with the financial side of the conference; and to Harold Esche and Rod Blais for providing us with a Web site at the University of Calgary. Finally, we would like to express our gratitude to our colleagues from the School of Computer Science at The Queen's University of Belfast and the Department of Computer Science at The University of Reading, who assisted in the organization of ICCS 2001.

May 2001

Vassil N. Alexandrov
Jack J. Dongarra
Benjoe A. Juliano
Reneé S. Renner
C. J. Kenneth Tan

Organization

The 2001 International Conference on Computational Science was organized jointly by The University of Reading (Department of Computer Science), The University of Tennesse (Department of Computer Science), and The Queen's University of Belfast (School of Computer Science).

Organizing Committee

Conference Chairs:	Vassil N. Alexandrov, *Department of Computer Science, The University of Reading* Jack J. Dongarra, *Department of Computer Science, University of Tennessee* C. J. Kenneth Tan, *School of Computer Science, The Queen's University of Belfast*
Local Organizing Chairs:	Benjoe A. Juliano (*California State University at Chico*, USA) Reneé S. Renner (*California State University at Chico*, USA)

Local Organizing Committee

Larry Davis (*Department of Defense HPC Modernization Program*, USA)
Benjoe A. Juliano *(California State University at Chico*, USA)
Cathy McDonald (*Department of Defense HPC Modernization Program*, USA)
Reneé S. Renner (*California State University at Chico*, USA)
C. J. Kenneth Tan (*The Queen's University of Belfast*, UK)
Valerie B. Thomas (*Department of Defense HPC Modernization Program*, USA)

Steering Committee

Vassil N. Alexandrov (*The University of Reading*, UK)
Marian Bubak (*AGH*, Poland)
Jack J. Dongarra (*Oak Ridge National Laboratory*, USA)
C. J. Kenneth Tan *(The Queen's University of Belfast*, UK)
Jerzy Waśniewski (*Danish Computing Center for Research and Education*, DK)

Special Events Committee

Vassil N. Alexandrov (*The University of Reading*, UK)
J. A. Rod Blais (*University of Calgary*, Canada)
Peter M. A. Sloot (*University of Amsterdam*, The Netherlands)
Marina L. Gavrilova (*University of Calgary*, Canada)

Program Committee

Vassil N. Alexandrov (*The University of Reading*, UK)
Hamid Arabnia (*University of Georgia*, USA)
J. A. Rod Blais (*University of Calgary*, Canada)
Alexander V. Bogdanov (*IHPCDB*)
Marian Bubak (*AGH*, Poland)
Toni Cortes (*Universidad de Catalunya, Barcelona*, Spain)
Brian J. d'Auriol (*University of Texas at El Paso*, USA)
Larry Davis (*Department of Defense HPC Modernization Program*, USA)
Ivan T. Dimov (*Bulgarian Academy of Science*, Bulgaria)
Jack J. Dongarra (*Oak Ridge National Laboratory*, USA)
Harold Esche (*University of Calgary*, Canada)
Marina L. Gavrilova (*University of Calgary*, Canada)
Ken Hawick (*University of Wales, Bangor*, UK)
Bob Hertzberger (*University of Amsterdam*, The Netherlands)
Michael J. Hobbs (*HP Labs, Palo Alto*, USA)
Caroline Isaac (*IBM UK*, UK)
Heath James (*University of Adelaide*, Australia)
Benjoe A. Juliano (*California State University at Chico*, USA)
Aneta Karaivanova (*Florida State University*, USA)
Antonio Laganà (*University of Perugia*, Italy)
Christiane Lemieux (*University of Calgary*, Canada)
Jiri Nedoma (*Academy of Sciences of the Czech Republic*, Czech Republic)
Cathy McDonald (*Department of Defense HPC Modernization Program*, USA)
Graham M. Megson (*The University of Reading*, UK)
Peter Parsons (*Sun Microsystems*, UK)
James S. Pascoe (*The University of Reading*, UK)
William R. Pulleyblank (*IBM T. J. Watson Research Center*, USA)
Andrew Rau-Chaplin (*Dalhousie University*, Canada)
Reneé S. Renner (*California State University at Chico*, USA)
Paul Roe (*Queensland University of Technology*, Australia)
Laura A. Salter (*University of New Mexico*, USA)
Peter M. A. Sloot (*University of Amsterdam*, The Netherlands)
David Snelling (*Fujitsu European Center for Information Technology*, UK)
Lois Steenman-Clarke (*The University of Reading*, UK)
C. J. Kenneth Tan (*The Queen's University of Belfast*, UK)
Philip Tannenbaum (*NEC/HNSX*, USA)
Valerie B. Thomas (*Department of Defense HPC Modernization Program*, USA)
Koichi Wada *University of Tsukuba*, Japan)
Jerzy Wasniewski (*Danish Computing Center for Research and Education*, DK)
Roy Williams (*California Institute of Technology*, USA)
Zahari Zlatev (*Danish Environmental Research Institute*, Denmark)
Elena Zudilova (*Corning Scientific Center*, Russia)

Sponsoring Organizations

American Mathematical Society, USA
Fujitsu European Center for Information Technology, UK
International Business Machines, USA
Pacific Institute for the Mathematical Sciences, Canada
Springer-Verlag, Germany
Sun Microsystems, USA
California State University at Chico, USA
The Queen's University of Belfast, UK
The University of Reading, UK

Table of Contents, Part I

Invited Speakers

Exploiting OpenMP to Provide Scalable SMP BLAS and LAPACK Routines
Cliff Addison. 3

Scientific Discovery through Advanced Computing
Carl Edward Oliver. 4

Quantification of Uncertainty for Numerical Simulations with Confidence Intervals
James Glimm. 5

Large-Scale Simulation and Visualization in Medicine: Applications to Cardiology, Neuroscience, and Medical Imaging
Christopher Johnson. 6

Can Parallel Programming Be Made Easy for Scientists?
Péter Kacsuk. 7

Software Support for High Performance Problem-Solving on Computational Grids
Ken Kennedy. 8

Lattice Rules and Randomized Quasi-Monte Carlo
Pierre L'Ecuyer. 9

Blue Gene: A Massively Parallel System
José E. Moreira. 10

Dynamic Grid Computing
Edward Siedel. 11

Robust Geometric Computation Based on Topological Consistency
Kokichi Sugihara. 12

Metacomputing with the Harness and IceT Systems
Vaidy Sunderam. 27

Computational Biology: IT Challenges and Opportunities
Stefan Unger, Andrew Komornicki. 28

Architecture-Specific Automatic Performance Tuning

A Data Broker for Distributed Computing Environments
L.A. Drummond, J. Demmel, C.R. Mechoso, H. Robinson, K. Sklower, J.A. Spahr. 31

Towards an Accurate Model for Collective Communications
Sathish Vadhiyar, Graham E. Fagg, and Jack J. Dongarra. . . . 41

A Family of High-Performance Matrix Multiplication Algorithms
John A. Gunnels, Greg M. Henry, Robert A. van de Geijn. . . . 51

Performance Evaluation of Heuristics for Scheduling Pipelined Multiprocessor Tasks
M. Fikret Ercan, Ceyda Oguz, Yu-Fai Fung. 61

Automatic Performance Tuning in the UHFFT Library
Dragan Mirković, S. Lennart Johnsson. 71

A Modal Model of Memory
Nick Mitchell, Larry Carter, Jeanne Ferrante. 81
Fast Automatic Generation of DSP Algorithms
Markus Püschel, Bryan Singer, Manuela Veloso, José M.F. Moura. 97
Cache-Efficient Multigrid Algorithms
Sriram Sellappa, Siddhartha Chatterjee. 107
Statistical Models for Automatic Performance Tuning
Richard Vuduc, James W. Demmel, Jeff Bilmes. 117
Optimizing Sparse Matrix Computations for Register Reuse in SPARSITY
Eun-Jin Im, Katherine Yelick. 127
Rescheduling for Locality in Sparse Matrix Computations
Michelle Mills Strout, Larry Carter, Jeanne Ferrante. 137

Climate Modeling
The DOE Parallel Climate Model (PCM): The Computational Highway and Backroads
Thomas Bettge, Anthony Craig, Rodney James, Vince Wayland, Gary Strand. 149
Conceptualizing a Collaborative Problem-Solving Environment for Regional Climate Modeling and Assessment of Climate Impacts
George Chin Jr., L. Ruby Leung, Karen Schuchardt, Debbie Gracio. 159
Computational Design and Performance of the Fast Ocean Atmosphere Model, Version 1
Robert Jacob, Chad Schafer, Ian Foster, Michael Tobis, John Anderson. 175
The Model Coupling Toolkit
J. Walter Larson, Robert L. Jacob, Ian T. Foster, Jing Guo. . . 185
Parallelization of a Subgrid Orographic Precipitation Scheme in an MM5-based Regional Climate Model
L. Ruby Leung, John G. Michalakes, Xindi Bian. 195
Resolution Dependence in Modeling Extreme Weather Events
John Taylor, Jay Larson. 204
Visualizing High-Resolution Climate Data
Sheri A. Voelz, John Taylor. 212

Global Computing – Internals and Usage
Improving Java Server Performance with Interruptlets
David Craig, Steven Carroll, Fabian Breg, Dimitrios S. Nikolopoulos, Constantine Polychronopoulos. 223
Protocols and Software for Exploiting Myrinet Clusters
P. Geoffray, C. Pham, L. Prylli, B. Tourancheau, R. Westrelin. 233
Cluster Configuration Aided by Simulation
Dieter F. Kvasnicka, Helmut Hlavacs, Christoph W. Ueberhuber. 243

Application Monitoring in the Grid with GRM and PROVE
Zoltán Balaton, Péter Kacsuk, Norbert Podhorszki. 253
Extension of Macrostep Debugging Methodology Towards Metacomputing Applications
Robert Lovas, Vaidy S. Sunderam. 263
Capacity and Capability Computing Using Legion
Anand Natrajan, Marty A. Humphrey, Andrew S. Grimshaw. . . 273
Component Object Based Single System Image Middleware for Metacomputer Implementation of Genetic Programming on Clusters
Ivan Tanev, Takashi Uozomi, Dauren Akhmetov. 284
The Prioritized and Distributed Synchronization in Distributed Groups
Michel Trehel, Ahmed Housni. 294

Collaborative Computing
On Group Communication Systems: Insight, a Primer and a Snapshot
P.A. Gray, J.S. Pascoe. 307
Overview of the InterGroup Protocols
K. Berket, D.A. Agarwal, P.M. Melliar-Smith, L.E. Moser. . . . 316
Introducing Fault-Tolerant Group Membership into the Collaborative Computing Transport Layer
R.J. Loader, J.S. Pascoe, V.S. Sunderam. 326
A Modular Collaborative Parallel CFD Workbench
Kwai L. Wong, A. Jerry Baker. 336
Distributed Name Service in Harness
Tomasz Tyrakowski, Vaidy S. Sunderam, Mauro Migliardi. 345
Fault Tolerant MPI for the Harness Meta-computing System
Graham E. Fagg, Antonin Bukovsky, Jack J. Dongarra. 355
A Harness Control Application for Hand-Held Devices
Tomasz Tyrakowski, Vaidy S. Sunderam, Mauro Migliardi. 367
Flexible Class Loader Framework: Sharing Java Resources in Harness System
Dawid Kurzyniec, Vaidy S. Sunderam. 375
Mobile Wide Area Wireless Fault-Tolerance
J.S. Pascoe, G. Sibley, V.S. Sunderam, R.J. Loader. 385
Tools for Collaboration in Metropolitan Wireless Networks
G. Sibley, V.S. Sunderam. 395
A Repository System with Secure File Access for Collaborative Environments
Paul A. Gray, Srividya Chandramohan, Vaidy S. Sunderam. 404
Authentication Service Model Supporting Multiple Domains in Distributed Computing
Kyung-Ah Chang, Byung-Rae Lee, Tai-Yun Kim. 413
Performance and Stability Analysis of a Message Oriented Reliable Multicast for Distributed Virtual Environments in Java
Gunther Stuer, Jan Broeckhove, Frans Arickx 423

A Secure and Efficient Key Escrow Protocol for Mobile Communications
Byung-Rae Lee, Kyung-Ah Chang, Tai-Yun Kim. 433

Complex Physical System Simulation
High-Performance Algorithms for Quantum Systems Evolution
Alexander V. Bogdanov, Ashot S. Gevorkyan, Elena N. Stankova. 447
Complex Situations Simulation when Testing Intelligence System Knowledge Base
Yu. I. Nechaev, A.B. Degtyarev, A.V. Boukhanovsky. 453
Peculiarities of Computer Simulation and Statistical Representation of Time-Spatial Metocean Fields
A.V. Boukhanovsky, A.B. Degtyarev, V.A. Rozhkov. 463
Numerical Investigation of Quantum Chaos in the Problem of Multichannel Scattering in Three Body System
A.V. Bogdanov, A.S. Gevorkyan, A.A. Udalov. 473
Distributed Simulation of Amorphous Hydrogenated Silicon Films: Numerical Experiments on a Linux Based Computing Environment
Yu.E. Gorbachev, M.A. Zatevakhin, V.V. Krzhizhanovskaya, A.A. Ignatiev, V. Kh. Protopopov, N.V. Sokolova, A.B. Witenberg. 483
Performance Prediction for Parallel Local Weather Forecast Programs
Wolfgang Joppich, Herrmann Mierendorff 492
The NORMA Language Application to Solution of Strong Nonequilibrium Transfer Processes Problem with Condensation of Mixtures on the Multiprocessors System
A.N. Andrianov, K.N. Efimkin, V. Yu. Levashov, I.N. Shishkova. 502
Adaptive High-Performance Method for Numerical Simulation of Unsteady Complex Flows with Number of Strong and Weak Discontinuities
Alexander Vinogradov, Vladimir Volkov, Vladimir Gidaspov, Alexander Muslaev, Peter Rozovski. 511
Cellular Automata as a Mesoscopic Approach to Model and Simulate Complex Systems
P.M.A. Sloot, A.G. Hoekstra. 518

Computational Chemistry
Ab-Initio Kinetics of Heterogeneous Catalysis: NO +N+ O/Rh(111)
A.P.J. Jansen, C.G.M. Hermse, F. Frechard, J.J. Lukkien. . . . 531
Interpolating Wavelets in Kohn-Sham Electronic Structure Calculations
A.J. Markvoort, R. Pino, P.A.J. Hilbers. 541
Simulations of Surfactant-Enhanced Spreading
Sean McNamara, Joel Koplik, Jayanth R. Banavar. 551

Supporting Car-Parrinello Molecular Dynamics Application with UNICORE
Valentina Huber. 560
Parallel Methods in Time Dependent Approaches to Reactive Scattering Calculations
Valentina Piermarini, Leonardo Pacifici, Stefano Crocchianti, Antonio Laganà, Giuseppina D'Agosto, Sergio Tasso. 567

Computational Finance
Construction of Multinomial Lattice Random Walks for Optimal Hedges
Yumi Yamada, James A. Primbs. 579
On Parallel Pseudo-random Number Generation
Chih Jeng Kenneth Tan. 589
A General Framework for Trinomial Trees
Ali Lari-Lavassani, Bradley D. Tifenbach. 597
On the Use of Quasi-Monte Carlo Methods in Computational Finance
Christiane Lemieux, Pierre L'Ecuyer. 607

Computational Geometry and Applications
An Efficient Algorithm to Calculate the Minkowski Sum of Convex 3D Polyhedra
Henk Bekker, Jos B.T.M. Roerdink. 619
REGTET: A Program for Computing Regular Tetrahedralizations
Javier Bernal. 629
Fast Maintenance of Rectilinear Centers
Sergei Bespamyatnikh, Michael Segal. 633
Exploring an Unknown Polygonal Environment with Bounded Visibility
Amitava Bhattacharya, Subir Kumar Ghosh, Sudeep Sarkar. 640
Parallel Optimal Weighted Links
Ovidiu Daescu. 649
Robustness Issues in Surface Reconstruction
Tamal K. Dey, Joachim Giesen, Wulue Zhao. 658
On a Nearest-Neighbor Problem in Minkowski and Power Metrics
M.L. Gavrilova. 663
On Dynamic Generalized Voronoi Diagrams in the Euclidean Metric
M.L. Gavrilova, J. Rokne. 673
Computing Optimal Hatching Directions in Layered Manufacturing
Man Chung Hon, Ravi Janardan, Jörg Schwerdt, Michiel Smid. . . 683
Discrete Local Fairing of B-spline Surfaces
Seok-Yong Hong, Chung-Seong Hong, Hyun-Chan Lee, Koohyun Park. 693
Computational Methods for Geometric Processing Applications to Industry
Andrés Iglesias, Akemi Gálvez, Jaime Puig-Pey. 698

Graph Voronoi Regions for Interfacing Planar Graphs
Thomas Kämpke, Matthias Strobel. 708
Robust and Fast Algorithm for a Circle Set Voronoi Diagram in a Plane
Deok-Soo Kim, Donguk Kim, Kokichi Sugihara, Joonghyun Ryu. 718
Apollonius Tenth Problem as a Point Location Problem
Deok-Soo Kim, Donguk Kim, Kokichi Sugihara, Joonghyun Ryu. 728
Crystal Voronoi Diagram and Its Applications to Collision-Free Paths
Kei Kobayashi, Kokichi Sugihara. 738
The Voronoi-Delaunay Approach for Modeling the Packing of Balls in a Cylindrical Container
V.A. Luchnikov, N.N. Medvedev, M.L. Gavrilova. 748
Multiply Guarded Guards in Orthogonal Art Galleries
T.S. Michael, Val Pinciu. 753
Reachability on a Region Bounded by Two Attached Squares
Ali Mohades, Mohammadreza Razzazi. 763
Illuminating Polygons with Vertex π-floodlights
Csaba D. Tóth. 772

Computational Methods

Performance Tradeoffs in Multi-tier Formulation of a Finite Difference Method
Scott B. Baden, Daniel Shalit. 785
On the Use of a Differentiated Finite Element Package for Sensitivity Analysis
Christian H. Bischof, H. Martin Bücker, Bruno Lang, Arno Rasch, Jakob W. Risch. 795
Parallel Factorizations with Algorithmic Blocking
Jaeyoung Choi. 802
Bayesian Parameter Estimation: A Monte Carlo Approch
Ray Gallagher, Tony Doran. 812
Recent Progress in General Sparse Direct Solvers
Anshul Gupta. 823
On Efficient Application of Implicit Runge-Kutta Methods to Large-Scale Systems of Index 1 Differential-Algebraic Equations
Gennady Yu. Kulikov, Alexandra A. Korneva. 832
On the Efficiency of Nearest Neighbor Searching with Data Clustered in Lower Dimensions
Songrit Maneewongvatana, David M. Mount. 842
A Spectral Element Method for Oldroyd-B Fluid in a Contraction Channel
Sha Meng, Xin Kai Li, Gwynne Evans. 852
SSE Based Parallel Solution for Power Systems Network Equations
Y.F. Fung, M. Fikret Ercan, T.K. Ho, W.L. Cheung. 862

Implementation of Symmetric Nonstationary Phase-Shift Wavefield Extrapolator on an Alpha Cluster
Yanpeng Mi, Gary F. Margrave. 874
Generalized High-Level Synthesis of Wavelet-Based Digital Systems via Nonlinear I/O Data Space Transformations
Dongming Peng, Mi Lu. 884
Solvable Map Method for Integrating Nonlinear Hamiltonian Systems
Govindan Rangarajan, Minita Sachidanand. 894
A Parallel ADI Method for a Nonlinear Equation Describing Gravitational Flow of Ground Water
I.V. Schevtschenko. 904
The Effect of the Cusp on the Rate of Convergence of the Rayleigh-Ritz Method
Ioana Sîrbu, Harry F. King. 911
The AGEB Algorithm for Solving the Heat Equation in Three Space Dimensions and Its Parallelization Using PVM
Mohd Salleh Sahimi, Norma Alias, Elankovan Sundararajan. . . 918
A Pollution Adaptive Mesh Generation Algorithm in r-h Version of the Finite Element Method
Soo Bum Pyun, Hyeong Seon Yoo. 928
An Information Model for the Representation of Multiple Biological Classifications
Neville Yoon, John Rose. 937
A Precise Integration Algorithm for Matrix Riccati Differential Equations
Wan-Xie Zhong, Jianping Zhu. 947

Computational Models of Natural Language Arguments
GEA: A Complete, Modular System for Generating Evaluative Arguments
Guiseppe Carenini. 959
Argumentation in Explanations to Logical Problems
Armin Fiedler, Helmut Horacek. 969
Analysis of the Argumentative Effect of Evaluative Semantics in Natural Language
Serge V. Gavenko. 979
Getting Good Value: Facts, Values and Goals in Computational Linguistics
Michael A. Gilbert. 989
Computational Models of Natural Language Argument
Chris Reed, Floriana Grasso. 999
An Empirical Study of Multimedia Argumentation
Nancy Green. 1009
Exploiting Uncertainty and Incomplete Knowledge in Deceptive Argumentation
Valeria Carofiglio, Fiorella de Rosis. 1019

Computational Physics in the Undergraduate Curriculum

Integrating Computational Science into the Physics Curriculum
Harvey Gould, Jan Tobochnik 1031

Musical Acoustics and Computational Science
N. Giordano, J. Roberts 1041

Developing Components and Curricula for a Research-Rich Undergraduate Degree in Computational Physics
Rubin H. Landau 1051

Physlets: Java Tools for a Web-Based Physics Curriculum
Wolfgang Christian, Mario Belloni, Melissa Dancy 1061

Computation in Undergraduate Physics: The Lawrence Approach
David M. Cook 1074

Computational Science Applications and Case Studies

Recent Developments of a Coupled CFD/CSD Methodology
Joseph D. Baum, Hong Luo, Eric L. Mestreau, Dmitri Sharov, Rainald Löhner, Daniele Pelessone, Charles Charman 1087

Towards a Coupled Environmental Prediction System
Julie L. McClean, Wieslaw Maslowski, Mathew E. Maltrud . . . 1098

New Materials Design
Jerry Boatz, Mark S. Gordon, Gregory Voth, Sharon Hammes-Shiffer, Ruth Pachter 1108

Parallelization of an Adaptive Mesh Refinement Method for Low Mach Number Combustion
Charles A. Rendleman, Vince E. Beckner, Mike J. Lijewski . . . 1117

Combustion Dynamics of Swirling Turbulent Flames
Suresh Menon, Vaidyanathan Sankaran, Christopher Stone . . . 1127

Parallel CFD Computing Using Shared Memory OpenMP
Hong Hu, Edward L. Turner 1137

Plasma Modeling of Ignition for Combustion Simulations
Osman Yaşar 1147

Computational Science Education: Standards, Learning Outcomes and Assessment Techniques

Computational Science Education: Standards, Learning Outcomes and Assessment
Osman Yaşar 1159

Learning Computational Methods for Partial Differential Equations from the Web
André Jaun, Johan Hedin, Thomas Johnson, Michael Christie, Lars-Erik Jonsson, Mikael Persson, Laurent Villard 1170

Computational Engineering and Science Program at the University of Utah
Carleton DeTar, Aaron L. Fogelson, Christopher R. Johnson, Christopher A. Sikorski 1176

High Performance and Parallel Computing in Manufacturing and Testing Environments

Influences on the Solution Process for Large, Numeric-Intensive Automotive Simulations
Myron Ginsberg. 1189

Salable Large Scale Process Modeling and Simulations in Liquid Composite Molding
Ram Mohan, Dale Shires, Andrew Mark. 1199

An Object-Oriented Software Framework for Execution of Real-Time, Parallel Algorithms
J. Brent Spears, Brett N. Gossage. 1209

A Multiagent Architecture Addresses the Complexity of Industry Process Re-engineering
John K. Debenham. 1219

Diagnosis Algorithms for a Symbolically Modeled Manufacturing Process
N. Rakoto-Ravalontsalama. 1228

Time-Accurate Turbine Engine Simulation in a Parallel Computing Environment: Part II - Software Alpha Test
M.A. Chappell, B.K. Feather. 1237

Monte Carlo Numerical Methods

Finding Steady State of Safety Systems Using the Monte Carlo Method
Ray Gallagher. 1253

Parallel High-Dimensional Integration: Quasi Monte-Carlo versus Adaptive Cubature Rules
Rudolf Schürer. 1262

Path Integral Monte Carlo Simulations and Analytical Approximations for High-Temperature Plasmas
V. Filinov, M. Bonitz, D. Kremp, W.-D. Kraeft, V. Fortov. . . . 1272

A Feynman-Kac Path-Integral Implementation for Poisson's Equation
Chi-Ok Hwang, Michael Mascagni. 1282

Relaxed Monte Carlo Linear Solver
Chih Jeng Kenneth Tan, Vassil Alexandrov. 1289

Author Index 1299

Table of Contents, Part II

Digital Imaging Applications

Densification of Digital Terrain Elevations Using Shape from Shading with Single Satellite Imagery 3
Mohammad A. Rajabi, J.A. Rod Blais

PC-Based System for Calibration, Reconstruction, Processing, and Visualization of 3D Ultrasound Data Based on a Magnetic-Field Position and Orientation Sensing System 13
Emad Boctor, A. Saad, Dar-Jen Chang, K. Kamel, A.M. Youssef

Automatic Real-Time XRII Local Distortion Correction Method for Digital Linear Tomography 23
Christian Forlani, Giancarlo Ferrigno

Meeting the Computational Demands of Nuclear Medical Imaging Using Commodity Clusters 27
Wolfgang Karl, Martin Schulz, Martin Völk, Sibylle Ziegler

An Image Registration Algorithm Based on Cylindrical Prototype Model . 37
Joong-Jae Lee, Gye-Young Kim, Hyung-Il Choi

An Area-Based Stereo Matching Using Adaptive Search Range and Window Size 44
Han-Suh Koo, Chang-Sung Jeong

Environmental Modeling

Methods of Sensitivity Theory and Inverse Modeling for Estimation of Source Term and Risk/Vulnerability Areas 57
Vladimir Penenko, Alexander Baklanov

The Simulation of Photochemical Smog Episodes in Hungary and Central Europe Using Adaptive Gridding Models 67
István Lagzi, Alison S. Tomlin, Tamás Turányi, László Haszpra, Róbert Mészáros, Martin Berzins

Numerical Solution of the Aerosol Condensation/Evaporation Equation . . . 77
Khoi Nguyen, Donald Dabdub

Efficient Treatment of Large-Scale Air Pollution Models on Supercomputers 82
Zahari Zlatev

High Performance Computational Tools and Environments

Pattern Search Methods for Use-Provided Points 95
Pedro Alberto, Fernando Nogueira, Humberto Rocha, Luís N. Vicente

In-situ Bioremediation: Advantages of Parallel Computing and Graphical Investigating Techniques 99
M.C. Baracca, G. Clai, P. Ornelli

Adaptive Load Balancing for MPI Programs 108
Milind Bhandarkar, L.V. Kalé, Eric de Sturler, Jay Hoeflinger

Performance and Irregular Behavior of Adaptive Task Partitioning 118
Elise de Doncker, Rodger Zanny, Karlis Kaugars, Laurentiu Cucos

Optimizing Register Spills for Eager Functional Languages 128
S. Mishra, K. Sikdar, M. Satpathy

A Protocol for Multi-threaded Processes with Choice in π-Calculus 138
Kazunori Iwata, Shingo Itabashi, Naohiro Ishi

Mapping Parallel Programs onto Distributed Computer Systems with Faulty Elements 148
Mikhail S. Tarkov, Youngsong Mun, Jaeyoung Choi, Hyung-Il Choi

Enabling Interoperation of High Performance, Scientific Computing Applications: Modeling Scientific Data with the Sets and Fields (SAF) Modeling System 158
Mark C. Miller, James F. Reus, Robb P. Matzke, William J. Arrighi, Larry A. Schoof, Ray T. Hitt, Peter K. Espen

Intelligent Systems Design and Applications

ALEC: An Adaptive Learning Framework for Optimizing Artificial Neural Networks 171
Ajith Abraham, Baikunth Nath

Solving Nonlinear Differential Equations by a Neural Network Method 181
Lucie P. Aarts, Peter Van der Veer

Fuzzy Object Blending in 2D 190
Ahmet Çinar, Ahmet Arslan

An Adaptive Neuro-Fuzzy Approach for Modeling and Control of Nonlinear Systems 198
Otman M. Ahtiwash, Mohd Zaki Abdulmui

The Match Fit Algorithm - A Testbed for Computational Motivation of Attention 208
Joseph G. Billock, Demetri Psaltis, Christof Koch

Automatic Implementation and Simulation of Qualitative Cognitive Maps . 217
João Paulo Carvalho, José Alberto Tomé

Inclusion-Based Approximate Reasoning 221
Chris Cornelis, Etienne E. Kerre

Attractor Density Models with Application to Analyzing the Stability of Biological Neural Networks 231
Christian Storm, Walter J. Freeman

MARS: Still an Alien Planet in Soft Computing? 235
Ajith Abraham, Dan Steinberg

Data Reduction Based on Spatial Partitioning 245
Gongde Guo, Hui Wang, David Bell, Qingxiang Wu
Alternate Methods in Reservoir Simulation 253
Guadalupe I. Janoski, Andrew H. Sung
Intuitionistic Fuzzy Sets in Intelligent Data Analysis for Medical Diagnosis 263
Eulalia Szmidt, Janusz Kacprzyk
Design of a Fuzzy Controller Using a Genetic Algorithm for Stator Flux Estimation 272
Mehmet Karakose, Mehmet Kaya, Erhan Akin
Object Based Image Ranking Using Neural Networks 281
Gour C. Karmakar, Syed M. Rahman, Laurence S. Dooley
A Genetic Approach for Two Dimensional Packing with Constraints 291
Wee Sng Khoo, P. Saratchandran, N. Sundararajan
Task Environments for the Dynamic Development of Behavior 300
Derek Harter, Robert Kozma
Wavelet Packet Multi-layer Perceptron for Chaotic Time Series Prediction: Effects of Weight Initialization 310
Kok Keong Teo, Lipo Wang, Zhiping Lin
Genetic Line Search 318
S. Lozano, J.J. Domínguez, F. Guerrero, K. Smith
HARPIC, an Hybrid Architecture Based on Representations, Perceptions, and Intelligent Control: A Way to Provide Autonomy to Robots 327
Dominique Luzeaux, André Dalgalarrondo
Hybrid Intelligent Systems for Stock Market Analysis 337
Ajith Abraham, Baikunth Nath, P.K. Mahanti
On the Emulation of Kohonen's Self-Organization via Single-Map Metropolis-Hastings Algorithms 346
Jorge Muruzábal
Quasi Analog Formal Neuron and Its Learning Algorithm Hardware 356
Karen Nazaryan
Producing Non-verbal Output for an Embodied Agent in an Intelligent Tutoring System 366
Roger Nkambou, Yan Laporte
Co-evolving a Neural-Net Evaluation Function for Othello by Combining Genetic Algorithms and Reinforcement Learning 377
Joshua A. Singer
Modeling the Effect of Premium Changes on Motor Insurance Customer Retention Rates Using Neural Networks 390
Ai Cheo Yeo, Kate A. Smith, Robert J. Willis, Malcolm Brooks
On the Predictability of Rainfall in Kerala - An Application of ABF Neural Network 400
Ninan Sajeeth Philip, K. Babu Joseph
A Job-Shop Scheduling Problem with Fuzzy Processing Times 409
Feng-Tse Lin

Speech Synthesis Using Neural Networks Trained by an Evolutionary Algorithm 419
Trandafir Moisa, Dan Ontanu, Adrian H. Dediu

A Two-Phase Fuzzy Mining and Learning Algorithm for Adaptive Learning Environment 429
Chang Jiun Tsai, S.S. Tseng, Chih-Yang Lin

Applying Genetic Algorithms and Other Heuristic Methods to Handle PC Configuration Problems 439
Vincent Tam, K.T. Ma

Forecasting Stock Market Performance Using Hybrid Intelligent System . . . 441
Xiaodan Wu, Ming Fung, Andrew Flitman

Multimedia

The MultiMedia Maintenance Management (M^4) System 459
Rachel J. McCrindle

Visualisations; Functionality and Interaction 470
Claire Knight, Malcolm Munro

DMEFS Web Portal: A METOC Application 476
Avichal Mehra, Jim Corbin

The Validation Web Site: A Combustion Collaboratory over the Internet . . 485
Angela Violi, Xiaodong Chen, Gary Lindstrom, Eric Eddings, Adel F. Sarofim

The Policy Machine for Security Policy Management 494
Vincent C. Hu, Deborah A. Frincke, David F. Ferraiolo

Multi-spectral Scene Generation and Projection

The Javelin Integrated Flight Simulation 507
Charles Bates, Jeff Lucas, Joe Robinson

A Multi-spectral Test and Simulation Facility to Support Missile Development, Production, and Surveillance Programs 515
James B. Johnson, Jerry A. Ray

Correlated, Real Time Multi-spectral Sensor Test and Evaluation (T&E) in an Installed Systems Test Facility (ISTF) Using High Performance Computing 521
John Kriz, Tom Joyner, Ted Wilson, Greg McGraner

Infrared Scene Projector Digital Model Development 531
Mark A. Manzardo, Brett Gossage, J. Brent Spears, Kenneth G. LeSueur

Infrared Scene Projector Digital Model Mathematical Description 540
Mark A. Manzardo, Brett Gossage, J. Brent Spears, Kenneth G. LeSueur

Distributed Test Capability Using Infrared Scene Projector Technology ... 550
David R. Anderson, Ken Allred, Kevin Dennen, Patrick Roberts, William R. Brown, Ellis E. Burroughs, Kenneth G. LeSueur, Tim Clardy
Development of Infrared and Millimeter Wave Scene Generators for the P3I BAT High Fidelity Flight Simulation ... 558
Jeremy R. Farris, Marsha Drake

Novel Models for Parallel Computation
A Cache Simulator for Shared Memory Systems ... 569
Florian Schintke, Jens Simon, Alexander Reinefeld
On the Effectiveness of D-BSP as a Bridging Model of Parallel Computation ... 579
Gianfranco Bilardi, Carlo Fantozzi, Andrea Pietracaprina, Geppino Pucci
Coarse Grained Parallel On-Line Analytical Processing (OLAP) for Data Mining ... 589
Frank Dehne, Todd Eavis, Andrew Rau-Chaplin
Architecture Independent Analysis of Parallel Programs ... 599
Ananth Grama, Vipin Kumar, Sanjay Ranka, Vineet Singh
Strong Fault-Tolerance: Parallel Routing in Networks with Faults ... 609
Jianer Chen, Eunseuk Oh
Parallel Algorithm Design with Coarse-Grained Synchronization ... 619
Vijaya Ramachandran
Parallel Bridging Models and Their Impact on Algorithm Design ... 628
Friedhelm Meyer auf der Heide, Rolf Wanka
A Coarse-Grained Parallel Algorithm for Maximal Cliques in Circle Graphs ... 638
E.N. Cáceres, S.W. Song, J.L. Szwarcfiter
Parallel Models and Job Characterization for System Scheduling ... 648
X. Deng, H. Ip, K. Law, J. Li, W. Zheng, S. Zhu

Optimization
Heuristic Solutions for the Multiple-Choice Multi-dimension Knapsack Problem ... 659
M. Mostofa Akbar, Eric G. Manning, Gholamali C. Shoja, Shahadat Khan
Tuned Annealing for Optimization ... 669
Mir M. Atiqullah, S.S. Rao
A Hybrid Global Optimization Algorithm Involving Simplex and Inductive Search ... 680
Chetan Offord, Željko Bajzer
Applying Evolutionary Algorithms to Combinatorial Optimization Problems ... 689
Enrique Alba Torres, Sami Khuri

Program and Visualization

Exploratory Study of Scientific Visualization Techniques for Program Visualization 701
Brian J. d'Auriol, Claudia V. Casas, Pramod K. Chikkappaiah, L. Susan Draper, Ammar J. Esper, Jorge López, Rajesh Molakaseema, Seetharami R. Seelam, René Saenz, Qian Wen, Zhengjing Yang

Immersive Visualization Using AVS/Express 711
Ian Curington

VisBench: A Framework for Remote Data Visualization and Analysis 718
Randy W. Heiland, M. Pauline Baker, Danesh K. Tafti

The Problem of Time Scales in Computer Visualization 728
Mark Burgin, Damon Liu, Walter Karplus

Making Movies: Watching Software Evolve through Visualisation 738
James Westland Chain, Rachel J. McCrindle

Tools and Environments for Parallel and Distributed Programming

Performance Optimization for Large Scale Computing: The Scalable VAMPIR Approach 751
Holger Brunst, Manuela Winkler, Wolfgang E. Nagel, Hans-Christian Hoppe

TRaDe: Data Race Detection for Java 761
Mark Christiaens, Koen De Bosschere

Automation of Data Traffic Control on DSM Architectures 771
Michael Frumkin, Haoqiang Jin, Jerry Yan

The Monitoring and Steering Environment 781
Christian Glasner, Roland Hügl, Bernhard Reitinger, Dieter Kranzlmüller, Jens Volkert

Token Finding Using Mobile Agents 791
Delbert Hart, Mihail E. Tudoreanu, Eileen Kraemer

Load Balancing for the Electronic Structure Program GREMLIN in a Very Heterogenous SSH-Connected WAN-Cluster of UNIX-Type Hosts 801
Siegfried Höfinger

DeWiz - Modular Debugging for Supercomputers and Computational Grids 811
Dieter Kranzlmüller

Fiddle: A Flexible Distributed Debugger Architecture 821
João Lourenço, José C. Cunha

Visualization of Distributed Applications for Performance Debugging 831
F.-G. Ottogalli, C. Labbé, V. Olive, B. de Oliveira Stein, J. Chassin de Kergommeaux, J.-M. Vincent

Achieving em Performance Portability with em SKaMPI for High-Performance MPI Programs 841
Ralf Reussner, Gunnar Hunzelmann
Cyclic Debugging Using Execution Replay 851
Michiel Ronsse, Mark Christiaens, Koen De Bosschere
Visualizing the Memory Access Behavior of Shared Memory Applications on NUMA Architectures 861
Jie Tao, Wolfgang Karl, Martin Schulz
CUMULVS Viewers for the ImmersaDesk 871
Torsten Wilde, James A. Kohl, Raymond E. Flanery

Simulation
N-Body Simulation on Hybrid Architectures 883
P.M.A. Sloot, P.F. Spinnato, G.D. van Albada
Quantum Mechanical Simulation of Vibration-Torsion-Rotation Levels of Methanol 893
Yun-Bo Duan, Anne B. McCoy
Simulation-Visualization Complexes as Generic Exploration Environment . 903
Elena V. Zudilova
Efficient Random Process Generation for Reliable Simulation of Complex Systems 912
Alexey S. Rodionov, Hyunseung Choo, Hee Y. Youn, Tai M. Chung, Kiheon Park
Replicators & Complementarity: Solving the Simplest Complex System without Simulation 922
Anil Menon

Soft Computing: Systems and Applications
More Autonomous Hybrid Models in Bang^2 935
Roman Neruda, Pavel Krušina, Zuzana Petrová
Model Generation of Neural Network Ensembles Using Two-Level Cross-Validation 943
S. Vasupongayya, R.S. Renner, B.A. Juliano
A Comparison of Neural Networks and Classical Discriminant Analysis in Predicting Students' Mathematics Placement Examination Scores 952
Stephen J. Sheel, Deborah Vrooman, R.S. Renner, Shanda K. Dawsey
Neural Belief Propagation without Multiplication 958
Michael J. Barber
Fuzzy Logic Basis in High Performance Decision Support Systems 965
A. Bogdanov, A. Degtyarev, Y. Nechaev
Scaling of Knowledge in Random Conceptual Networks 976
Lora J. Durak, Alfred W. Hübler

Implementation of Kolmogorov Learning Algorithm for Feedforward Neural Networks 986
Roman Neruda, Arnošt Štědrý, Jitka Drkošová
Noise-Induced Signal Enhancement in Heterogeneous Neural Networks 996
Michael J. Barber, Babette K. Dellen

Phylogenetic Inference for Genome Rearrangement Data
Evolutionary Puzzles: An Introduction to Genome Rearrangement 1003
Mathieu Blanchette
High-Performance Algorithmic Engineering for Computationa Phylogenetics 1012
Bernard M.E. Moret, David A. Bader, Tandy Warnow
Phylogenetic Inference from Mitochondrial Genome Arrangement Data . . 1022
Donald L. Simon, Bret Larget

Late Submissions
Genetic Programming: A Review of Some Concerns 1031
Maumita Bhattacharya, Baikunth Nath
Numerical Simulation of Quantum Distributions: Instability and Quantum Chaos 1041
G.Y. Kryuchkyan, H.H. Adamyan, S.B. Manvelyan
Identification of MIMO Systems by Input-Output Takagi-Sugeno Fuzzy Models 1050
Nirmal Singh, Renu Vig, J.K. Sharma
Control of Black Carbon, the Most Effective Means of Slowing Global Warming 1060
Mark Z. Jacobson
Comparison of Two Schemes for the Redistribution of Moments for Modal Aerosol Model Application 1061
U. Shankar, A.L. Trayanov
A Scale-Dependent Dynamic Model for Scalar Transport in the Atmospheric Boundary Layer 1062
Fernando Port-Agel, Qiao Qin

Advances in Molecular Algorithms
MDT - The Molecular Dynamics Test Set 1065
Eric Barth
Numerical Methods for the Approximation of Path Integrals Arising in Quantum Statistical Mechanics 1066
Steve D. Bond
The Multigrid *N*-Body Solver 1067
David J. Hardy

Do Your Hard-Spheres Have Tails? A Molecular Dynamics Integration Algorithm for Systems with Mixed Hard-Core/Continuous Potentials 1068
Brian B. Laird
An Improved Dynamical Formulation for Constant Temperature and Pressure Dynamics, with Application to Particle Fluid Models 1069
Benedict J. Leimkuhler

Author Index 1071

Exploiting OpenMP to Provide Scalable SMP BLAS and LAPACK Routines

Cliff Addison

Research Manager
Fujitsu European Centre for
Information Technology Ltd.
2 Longwalk Road
Stockley Park, Uxbridge
Middlesex, England UB11 1AB
Phone: +44-(208)-606-4518
FAX: +44-(208)-606-4422
addison@fecit.co.uk

Abstract

The present Fujitsu PRIMEPOWER 2000 system can have up to 128 processors in an SMP node. It is therefore desirable to provide users of this system with high performance parallel BLAS and LAPACK routines that scale to as many processors as possible. It is also desirable that users can obtain some level of parallel performance merely by relinking their codes with SMP Math Libraries.

This talk outlines the major design decisions taken in providing OpenMP versions of BLAS and LAPACK routines to users, it discusses some of the algorithmic issues that have been addressed and it discusses some of short comings of OpenMP for this task.

A good deal has been learned about exploiting OpenMP in this on-going activity and the talk will attempt to identify what worked and what did not work. For instance, while OpenMP does not support recursion, some of the basic ideas behind linear algebra with recursive algorithms can be exploited to overlap sequential operations with parallel ones. As another example, the overheads of dynamic scheduling tended to outweigh the better load balancing that such a schedule provides so that static cyclic loop scheduling was more effective.

V.N. Alexandrov et al. (Eds.): ICCS 2001, LNCS 2073, p. 3, 2001.
© Springer-Verlag Berlin Heidelberg 2001

Scientific Discovery through Advanced Computing

Carl Edward Oliver

Associate Director of
Science for the Office of Advanced Scientific Computing Research
U. S. Department of Energy, SC-30
19901 Germantown Road
Germantown, Maryland 20874-1290
Phone: +1-(301)-903-7486
FAX: +1-(301)-903-4846
ed.oliver@science.doe.gov

Abstract

Scientific Discovery through Advanced Computing (SciDAC), a new initiative in the Department of Energy's Office of Science, will be described. Computational modeling and simulation are essential to all of the programs in the Office of Science and each of the programs has identified major scientific challenges that can only be addressed through advances in scientific computing.

Advances in computing technologies during the past decade have set the stage for significant advances in modeling and simulation in the coming decade. Several computer vendors promise to increase "peak" performance a 1000-fold in the next five years. Our challenge is to make similar advances in the scientific codes so performance does not degrade as the number of processors increases. This translates to increased investments in algorithms, tools, networking, system software, and applications software. Large interdisciplinary teams of applied mathematicians, computer scientists, and computational scientists are being formed to tackle this daunting problem.

These teams will be supported by a Scientific Computing Hardware Infrastructure designed to meet the needs of the Office of Science's research programs. It will be robust-to provide a reliable source of computing resources for scientific research; agile-to respond to innovative advances in computer technology; and flexible-to ensure that the most effective and efficient resources are used to solver each class of problems.

A status of SciDAC in its initial year and view of where we would like to be in five years will be presented.

V.N. Alexandrov et al. (Eds.): ICCS 2001, LNCS 2073, p. 4, 2001.
© Springer-Verlag Berlin Heidelberg 2001

Quantification of Uncertainty for Numerical Simulations with Confidence Intervals

James Glimm

Distinguished/Leading Professor
Dept of Applied Mathematics and Statistics
P-138A Math Tower
University at Stony Brook
Stony Brook, NY 11794-3600
Phone: +1-(516)-632-8355
FAX: +1-(516)-632-8490
glimm@ams.sunysb.edu

Abstract

We present a prediction and uncertainty assessment methodology for numerical simulation. The methodology allows prediction of confidence intervals. It has been developed jointly with a number of colleagues. It is a work in progress in the sense that not all components of the methodology are complete. The methodology, at its present level of deveopment, will be illustrated in two specific cases: the flow of oil in petroleum reservoirs (with prediction of production rates) and an analysis of solution errors for the simulation of shock wave interactions.

The formalism assesses uncertainty and yields confidence intervals associated with its prediction. In the terminology of verification and validation, these predictions can be verified as exact within a framework for statistical inference, but they are not validated as being descriptive of a physical situation. In fact the present illustrative examples are simplified not intended to represent an experimental or engineering system.

The methodology combines new developments in the traditional areas of oil reservoir upscaling and history matching with a new theory for numerical solution errors and with Bayesian inference. For the shock wave simulations, the new result is an error analysis for simple shock wave interactions.

The significance of our methods, in the petroleum reservoir context, is their ability to predict the risk, or uncertainty associated with production rate forecasts, and not just the production rates themselves. The latter feature of this method, which is not standard, is useful for evaluation of decision alternatives. For shock wave interactions, the significance of the methodology will be to contribute to verification and validation of simulation codes.

V.N. Alexandrov et al. (Eds.): ICCS 2001, LNCS 2073, p. 5, 2001.
© Springer-Verlag Berlin Heidelberg 2001

Large-Scale Simulation and Visualization in Medicine: Applications to Cardiology, Neuroscience, and Medical Imaging

Chris Johnson

Director, Scientific Computing and Imaging Institute
School of Computing
Merrill Engineering Building
50 South Campus Central Dr., Room 3490
University of Utah
Salt Lake City, Utah 84112-9205
Phone: +1-(801)-585-1867
FAX: +1-(801)-585-6513
crj@cs.utah.edu

Abstract

Computational problems in medicine often require a researcher to apply diverse skills in confronting problems involving very large data sets, three-dimensional complex geometries which must be modeled and visualized, large scale computing, and hefty amounts of numerical analysis. In this talk I will present recent research results in computational neuroscience, imaging, and cardiology. I will provide examples of several driving applications of steering and interactive visualization in cardiology (defibrillation simulation and device design), neuroscience (new inverse source localization techniques), and imaging (new methods for interactive visualization of large-scale 3D MRI and CT volumes, and introduce new methods for diffusion tensor imaging).

V.N. Alexandrov et al. (Eds.): ICCS 2001, LNCS 2073, p. 6, 2001.
© Springer-Verlag Berlin Heidelberg 2001

Can Parallel Programming Be Made Easy for Scientists?

Peter Kacsuk

Distinguished/Leading Professor MTA SZTAKI Research Institute
H-1132 Budapest Victor Hugo 18-22. Hungary
Phone: +36-(1)-329-7864 FAX: +36-(1)-329-7864
kacsuk@sztaki.hu

Abstract

The general opinion is that parallel programming is much harder than sequential programming. It is true if the progammer would like to reach over 90

Our P-GRADE environment was designed to meet these natural requirements of scientists. It is a completely graphical environment that supports the whole life-cycle of parallel program development. The programming language, called GRAPNEL, is a graphical extension of C, C++ or FORTRAN where graphics is used to express activities related to parallelism (like process creation, communication, etc.) and at the same time graphics hides the low level details of message passing library calls like PVM and MPI calls. Program constructs independent of parallelism can be inherited from sequential C, C++ or FORTRAN code. Moreover complete sequential C, C++ or FORTRAN libraries can be used in the GRAPNEL program and in this way parallelizing sequential code becomes extremely easy. Usage of predefined process topology templates enables the user to quickly generate very large parallel programs, too.

A user-friendly dragg-and-drop style graphical editor (GRED) helps the programmer to generate any necessary graphical constructs of GRPNEL. The DIWIDE distributed debugger provides systematic and automatic discovery of deadlock situations that are the most common problems of message passing parallel programs. DIWIDE also supports replay technique and hence the cyclic debugging techniques like breakpoint, step-by-step execution can be applied even in a non-deterministic parallel programming system. Performance analysis is supported by the GRM monitor and the PROVE execution visualization tool. The instrumentation is completely automatic, filters can be easily added or removed for the GRM monitor. The execution visualization can be done both off-line and on-line providing various synchronized trace-event views as well as statistics windows on processor utilization and communications. The connection between the source code and the trace-events can be easily identified by the source code click-back and click-forward facilities. GRM and PROVE are able to support the observation of real-size, long-running parallel programs, too. In many cases performance bottlenecks are due to wrong mapping of processes to processors. An easy-to-use mapping tool supports the user to quickly rearrange the processes on the processors of the parallel system.

The talk will highlight those features of P-GRADE that makes parallel programming really easy for non-hacker programmers, including scientists

V.N. Alexandrov et al. (Eds.): ICCS 2001, LNCS 2073, p. 7, 2001.
© Springer-Verlag Berlin Heidelberg 2001

Software Support for High Performance Problem-Solving on Computational Grids

Ken Kennedy

John Doerr Professor
Director, Center for High Performance Software
Computer Science MS132
3079 Duncan Hall
Rice University
Houston TX 77251-1892, USA
Phone: +1-(713)-348-5186
FAX: +1-(713)-348-5186
ken@rice.edu

Abstract

The 1999 report of the President's Information Technology Advisory Committee (PITAC)-Information Technology Research: Investing in our Future-called on the Federal government and the research community to shift their focus toward long-term, high-risk projects. This report has had a pronounced impact both on the structure of funding programs and on how we think about the entire IT research endeavor. One outcome is that researchers now think about their work in the context of some overarching effort of which it is a part. As a result, many more of us are thinking about long-term goals for IT research.

One extremely challenging problem for the coming decade is how to make it easy to develop applications for collections of heterogeneous, geographically-distributed computing platforms, sometimes called computational grids. In other words, how can we make the Internet a suitable computing platform for ordinary users? This talk will discuss the Grid Application Development Software (GrADS) Project, an effort funded by the NSF Next Generation Software Program, which is seeking to develop software strategies to simplify the problem of programming for a grid.

The GrADS effort is focusing on two challenges. First, how can we support the development of configurable object programs that can be retargeted to different collections of computing platforms and tailored for efficient execution once the target configuration is known? Second, how can we provide abstract interfaces to shield the average users from the complexities of programming for a network environment? One way to address this second problem is to make it possible for end users to develop programs in high-level domain-specific programming systems. I will discuss a new compiler framework, called telescoping languages, designed to make it easy to construct domain-specific scripting languages that achieve high performance on a variety of platforms including grids.

V.N. Alexandrov et al. (Eds.): ICCS 2001, LNCS 2073, p. 8, 2001.
© Springer-Verlag Berlin Heidelberg 2001

Lattice Rules and Randomized Quasi-Monte Carlo

Pierre L'Ecuyer

Département d'Informatique et de Recherche Opérationnelle
Université de Montréal
C.P.6128, Succ. Centre-Ville
Montréal, Québec H3C 3J7, Canada
Phone: +1-(514)-343-2143
FAX: +1-(514)-343-5834
lecuyer@iro.umontreal.ca

Abstract

High-dimensional multivariate integration is a difficult problem for which the Monte Carlo method is often the only viable approach. This method provides an unbiased estimator of the integral, together with a probabilistic error estimate (e.g., in the form of a confidence interval). The aim of randomized quasi-Monte Carlo (QMC) methods is to provide lower-variance unbiased estimators, also with error estimates. This talk will concentrate on one class of randomized QMC methods: randomized lattice rules.

We will explain how these methods fit into QMC methods in general and why they are interesting, how to choose their parameters, and how they can be used for medium and large-scale simulations. Numerical examples will be given to illustrate their effectiveness.

V.N. Alexandrov et al. (Eds.): ICCS 2001, LNCS 2073, p. 9, 2001.
© Springer-Verlag Berlin Heidelberg 2001

Blue Gene: A Massively Parallel System

Jose E. Moreira

Research Staff Member
IBM T. J. Watson Research Center
Yorktown Heights NY 10598-0218
phone: +1-(914)-945-3987
fax: +1-(914)-945-4425
jmoreira@us.ibm.com

Abstract

Blue Gene is a massively parallel system being developed at the IBM T. J. Watson Research Center. With its 4 million-way parallelism and 1 Petaflop peak performance, Blue Gene is a unique environment for research in parallel processing. Full exploitation of the machine's capability requires 100-way shared memory parallelism inside a single-chip multiprocessor node and message-passing across 30,000 nodes. New programming models, languages, compilers, and libraries will need to be investigated and developed for Blue Gene, therefore offering the opportunity to break new ground in those areas. In this talk, I will describe some of the hardware and software features of Blue Genes. I will also describe some of the protein science and molecular dynamics computations that are important driving forces behind Blue Gene.

V.N. Alexandrov et al. (Eds.): ICCS 2001, LNCS 2073, p. 10, 2001.
© Springer-Verlag Berlin Heidelberg 2001

Dynamic Grid Computing

Edward Siedel

Max-Planck-Institut für Gravitationsphysik
Albert-Einstein-Institut
Haus 5
Am Muehlenberg
14476 Golm, Germany
Phone: +49-(331)-567-7210
FAX: +49-(331)-567-7298
eseidel@aei-potsdam.mpg.de

Abstract

The Grid has the potential to fundamentally change the way science and engineering are done. Aggregate power of computing resources connected by networks—of the Grid— exceeds that of any single supercomputer by many orders of magnitude. At the same time, our ability to carry out computations of the scale and level of detail required, for example, to study the Universe, or simulate a rocket engine, are severely constrained by available computing power. Hence, such applications should be one of the main driving forces behind the development of Grid computing. I will discuss some large scale applications, including simulations of colliding black holes, and show how they are driving the development of Grid computing technology. Applications are already being developed that are not only aware of their needs, but also of the resources available to them on the Grid. They will be able to adapt themselves automatically to respond to their changing needs, to spawn off tasks on other resources, and to adapt to the changing characteristics of the Grid including machine and network loads and availability. I will discuss a number of innovative scenarios for computing on the Grid enabled by such technologies, and demonstrate how close these are to being a reality.

V.N. Alexandrov et al. (Eds.): ICCS 2001, LNCS 2073, p. 11, 2001.
© Springer-Verlag Berlin Heidelberg 2001

Robust Geometric Computation Based on Topological Consistency

Kokichi Sugihara

Department of Mathematical Information Science and Technology,
University of Tokyo,
7-3-1 Hongo, Bunkyo-ku, Tokyo 113-0033, Japan
sugihara@simplex.t.u-tokyo.ac.jp

Abstract. The paper surveys a method, called the "topology-oriented method", for designing numerically robust geometric algorithms. In this method, higher priority is placed on the consistency of the topological structures of geometric objects than on numerical values. The resulting software is completely robust in the sense that inconsistency never arizes no matter how large numerical errors take place. The basic idea of this method and typical examples are shown.

1 Introduction

Quite a large number of "efficient" algorithms have been proposed to solve geometric problems. However, those algorithms are fragile in general in the sense that, if we implement them naively, they easily fail due to numerical errors [31,7,5,15,16]. Theoreticians design algorithms on the assumption that there is no numerical error or degeneracy, but in real computation both numerical errors and degeneracy arise frequently. This gap between the ideal world and the real world causes a serious problem of instability in actual geometric computation.

To overcome this difficulty, many approaches have been proposed. To simplify the situation, we can classify these approaches into three groups according to how much they rely on numerical computation. These three groups are shown in Fig. 1. The horizontal axis in this figure represents the amount of reliability of numerical values assumed in the design of robust algorithms; numerical values are more reliable in the right than in the left.

The first group is the "exact-computation approach", in which numerical computation are carried out in sufficiently high precision [41,29,21,23,24,35,30,1] [10,40,45]. The topological structure of a geometric object can be decided by the signs of the results of numerical computations. If we restrict the precision of the input data, these signs can be judged correctly in a sufficiently high but still finite precision. Using this principle, the topological structures are judged correctly as if the computation is done exactly. In this approach, we need not worry about misjudgement and hence theoretical algorithms can be implemented rather straightforward.

In this approach, degenerate situations are recognized exactly, and hence exceptional branches of processing for degenerate cases are necessary to complete

V.N. Alexandrov et al. (Eds.): ICCS 2001, LNCS 2073, pp. 12–26, 2001.
© Springer-Verlag Berlin Heidelberg 2001

small	**reliance values numerical values**	**large**
⟵		⟶
topology-oriented approach	**tolerance-approach** ε**-geometry interval-arithmetic**	**exact-computation approach**

Fig. 1. Three groups of approaches to robustness.

the algorithms. However, such exceptional branches can be avoided by a symbolic perturbation scheme [6,35,44].

Another disadvantages of this approach is the computation cost. The computation in this approach is expensive, because multiple precisions are used. To decrease the cost, acceleration schemes are also considered. A typical method is a lazy evaluation scheme, in which computation is first done in floating-point arithmetic, and if the precision turns out to be insufficient, then they switch to multiple precision [1,4,10,32,40]. Another method is the use of modular arithmetic instead of the multiple precision [2,3,18].

The second group of approaches relies on numerical computation moderately. They start with the assumption that numerical computation contains errors but the amount of the errors is bounded. Every time numerical computation is done, the upper bound of the error is also evaluated. On the basis of this error bound, the result of computation is judged to be either reliable or unreliable, and the only reliable result is used [25,8,9,12,16,31,34]. This approach might be natural for programmers in order to cope with numerical errors, but it makes program codes unnecessarily complicated because every numerical computation should be followed by two alternative branches of processing, one for the reliable case and the other for the unreliable case. Moreover, this approach decreases the portability of the software products, because the amount of errors depends on computation environment.

The third group of approaches is the "topology-oriented approach", which does not rely on numerical computation at all. In this approach, we start with the assumption that every numerical computation contains errors and that the amount of the error cannot be bounded. We place the highest priority on the consistency of topological properties, and use numerical results only when they are consistent with the topological properties, thus avoiding inconsistency [42,43,20] [28,17,22,26,27,38,39].

In this paper, we concentrate on the third approach, i.e., the topology-oriented approach, and survey the basic idea and several examples.

2 Instability in Geometric Computation

First, we will see by an example how unstable the geometric computation is. Suppose that we are given a convex polyhedron Π and a plane H, and that

we want to cut the polyhedron by the plane and to take one part off. For this purpose we need to classify the vertices of Π into two groups, the vertices above H and those below H. Consider the situation where H is very close to and almost parallel to one face of Π, as shown in Fig. 2(a). Then, the classification of the vertices are easily violated by numerical errors. Hence, it can happen that, as shown in Fig. 2(b), a pair of mutually opposite vertices on the face are judged below H and the other vertices on the face are judged above H. This situation is inconsistent, because this situation implies that the face should meet H at two lines, while in Euclidean geometry two distinct planes can meet at most at one line. Such an inconsistent classification of the vertices usually causes the algorithm to fail.

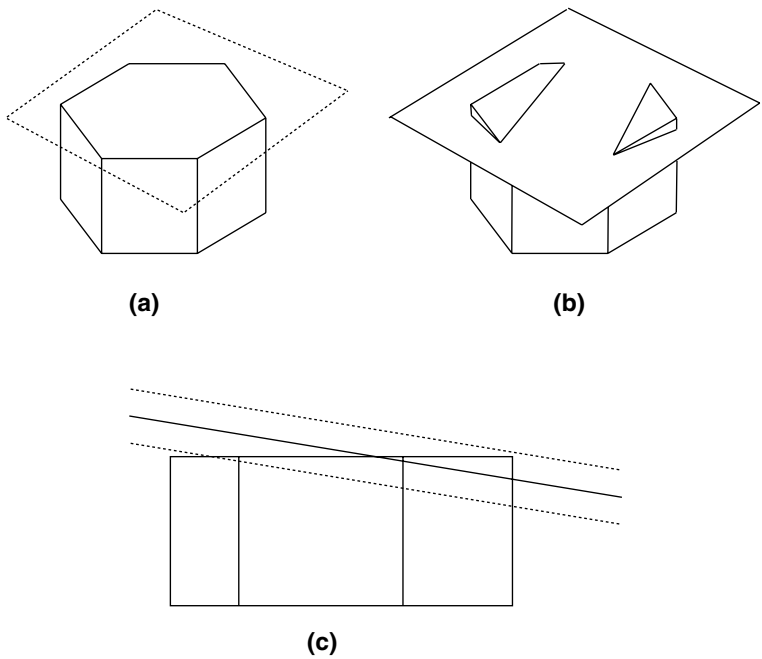


Fig. 2. Inconsistency caused by numerical errors in cutting a polyhedron by a plane.

A conventional method to circumvent this difficulty is to fix a certain small number ε, call a *tolerance*, and to consider two geometric elements to be at the same position if their distance is smaller than ε. Indeed, inconsistency can be avoided in many cases by this method. However, this is not complete; inconsistency still can happen.

Fig. 2(c) shows an example where the above method does not work. This is the picture of the scene in Fig. 1(a) seen in the direction parallel both to the top face of Π and the cut plane H. The pair of broken lines shows the region in

which the distance to H is smaller than ε. In this particular example, five of the vertices on the top face are judged to be exactly on H whereas the other vertex is judged below H. This is a contradiction, because in Euclidean geometry three or more noncollinear points being on H implies that the other vertices are also on H; hence all the vertices should be on the cut plane H.

3 Robustness and Consistency

Let P be a geometric problem, and let f be a theoretical algorithm to solve P. By a "theoretical" algorithm, we mean an algorithm that is designed assuming precise arithmetic, namely, one whose correctness is based on the assumption that no numerical error takes place in the computation.

The algorithm f can be considered a mapping from the set $\Xi(P)$ of all possible inputs to the set $\Omega(P)$ of all possible outputs. Each input $X \in \Xi(P)$ represents an instance of the problem P, and the corresponding output $f(X) \in \Omega(P)$ is a solution of the problem instance.

Both the input and the output can be divided into the "combinatorial and/or topological part" ("topological part" for short) and the "metric part." We represent the topological part by a subscript T and the metric part by a subscript M. More specifically, the input X is divided into the topological part X_{T} and the metric part X_{M}, and the output $f(X)$ is divided into the topological part $f_{\mathrm{T}}(X)$ and the metric part $f_{\mathrm{M}}(X)$.

For example, suppose that P is the problem of cutting a convex polyhedron by aplane. Then the topological part X_{T} of the input consists of the incidence relations among the vertices, the edges and the faces of the given polyhedron, and the metric part X_{M} consists of the equation of the cutting plane and the list of the three-dimensional coordinates of the vertices and/or the list of equations of the planes containing the faces. The topological part $f_{\mathrm{T}}(X)$ of the output consists of the incidence relations among the vertices, the edges and the faces of the computed polyhedron, and the metric part $f_{\mathrm{M}}(X)$ of the output consists of the list of the three-dimensional coordinates of the vertices of the computed polyhedron.

For another example, suppose that P is the problem of constructing the Voronoi diagram for a finite number of given points in the plane. Then the topological part X_{T} of the input consists of a single integer to represent the number N of points, and the metric part X_{M} is the set of the n pairs of coordinates of the points: $X_{\mathrm{T}} = \{n\}$ and $X_{\mathrm{M}} = \{x_1, y_1, \ldots, x_n, y_n\}$. The topological part $f_{\mathrm{T}}(X)$ of the output is the planar graph structure consisting of the Voronoi vertices and the Voronoi edges, and the metric part $f_{\mathrm{M}}(X)$ consists of the coordinates of the Voronoi vertices and the directions of the infinite Voronoi edges.

Let $\widetilde{f}$ denote an actually implemented computer program to solve P. The program $\widetilde{f}$ may be a simple translation of the algorithm f into a programming language, or it may be something more sophisticated aiming at robustness. The program $\tilde{f}$ can also be considered a mapping from the input set to the output

set. However, in actual situations, the program runs in finite-precision arithmetic, and consequently the behavior of $\widetilde{f}$ is usually different from that of f.

The program $\widetilde{f}$ is said to be *numerically robust* (or *robust* for short) if $\widetilde{f}(X)$ is defined for any input X in $\Xi(P)$. In other words, $\widetilde{f}$ is robust if it defines a total (not partial) function from $\Xi(P)$ to a superset $\widetilde{\Omega}(P)$ of $\Omega(P)$, i.e., if the program always carries out the task, ending up with some output, never entering into an endless loop nor terminating abnormally.

The program $\widetilde{f}$ is said to be *topologically consistent* (or *consistent* for short) if $\widetilde{f}$ is robust and $\widetilde{f}_{\mathrm{T}}(X) \in \Omega_{\mathrm{T}}(P)$ for any $X \in \Xi(P)$. In other words, $\widetilde{f}$ is consistent if the topological part $\widetilde{f}_{\mathrm{T}}(X)$ of the output coincides with the topological part $f_{\mathrm{T}}(X')$ of the correct solution of some instance X' (not necessarily equal to X) of the problem P.

Our goal is to construct $\widetilde{f}$ that is at least robust and hopefully consistent.

4 Basic Idra of the Topology-Oriented Approach

4.1 Basic Idea

In this section we suppose that exact arithmetic is not available and hence numerical computation contains errors. Fig. 3(a) shows how a conventional algorithm fails. Let $S = \{J_1, J_2, \ldots, J_n\}$ be the set of all the predicates that should be checked in the algorithm. Whether those predicates are true or not is judged on the basis of numerical computations. Since numerical computations contain errors, some of the predicates may be judged incorrectly, which in tern generate inconsistency and the algorithm fails.

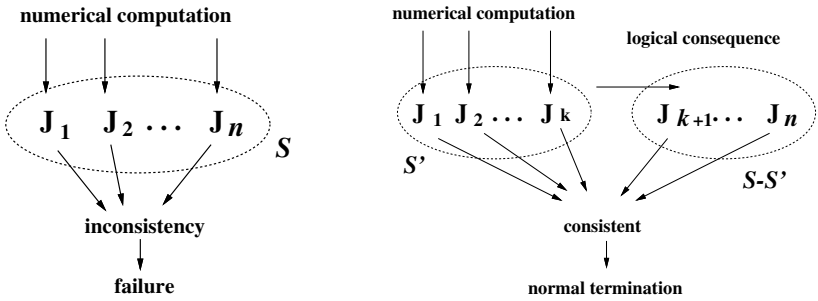


Fig. 3. Basic idea of the topology-oriented approach.

Numerical errors are inevitable in computation, but still we want to avoid inconsistency. To this goal, we first try to find a maximal subset, say S', of predicates that are independent from each other, as shown in Fig. 3(b), where "independent" means that the truth values of any predicates in S' do not affect

the truth values of the other predicates in this subset. The other predicates are dependent in the sense that their truth values are determined as the logical consequence of the truth values of the predicates in S'.

Once we find such a subset S', we evaluate the predicates in S' by numerical computation, and adopt the logical consequences of them as the truth values of the other predicates, i.e., the predicates in $S - S'$. Since the predicates in S' are independent, any assignment of the truth values to the predicates in S' does not generate inconsistency. Moreover, since we adopt the logical consequences of these truth values as the truth values of the predicates in $S - S'$, we never come across inconsistency.

We cannot guarantee the correctness of the truth values in S' because we have numerical errors, but once we believe the results of numerical computations, we can construct a consistent world. This is the basic idea for avoiding inconsistency. In the following subsections we will show how this idea works using typical example problems.

5 Examples

5.1 Cutting a Convex Polyhedron by a Plane

Let Π be a convex polyhedron in a three-dimensional space, and H be a plane. We consider the problem of cutting Π by H and taking one part off. Theoretically this problem is not difficult. What we have to do is to classify the vertices of Π into those above H and those below H. Once we classify them, we can determine the topological structure of the resulting polyhedron.

However, a naive implementation of this algorithm is not robust, as we have already seen in section 2.

To attain numerical robustness, we concentrate on the topological part of the algorithm. From the topological point of view, the vertices and the edges of Π form a planar graph, say G. As shown in Fig. 4, to cut Π by H, we first find the vertices that are above H (the vertices with black circles in Fig. 4(b)), next generate new vertices on the edges connecting the vertices above H and those below H (the vertices with white circles in Fig. 4(b)), generate a new cycle connecting them (the broken lines in Fig. 4(b)), and finally remove the substructure inside the cycle (Fig. 4(c)).

Let V_1 be the set of vertices of G that are judged above H, and let V_2 be the set of vertices that are judged below H. Since Π is convex, the next property holds.

Proposition 1. The subgraph of G induced by V_1 and that induced by V_2 are both connected.

We place higher priority on this property; we employ numerical results only when they do not contradict against this property. In this way we can construct a numerically robust algorithm [38].

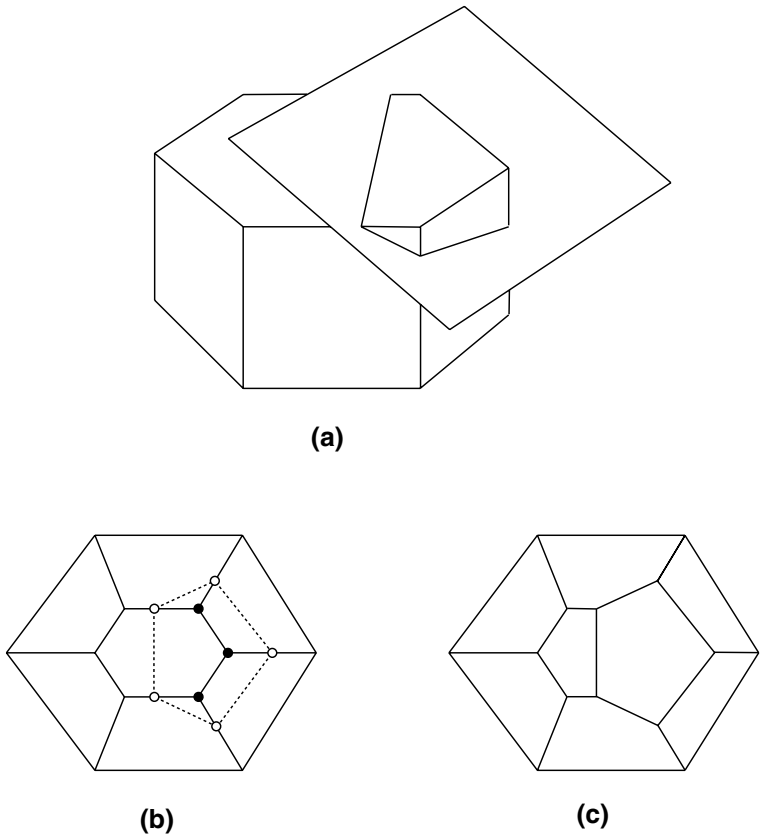


Fig. 4. Topological aspect of the cut operation.

An example of the behavior of this algorithm is shown in Fig. 5. Fig. 5(a) is the output of the algorithm when a cube is cut by 500 planes that are tangent to a common sphere. This problem is not difficult; a naively implemented software may also be able to give the same output. However, our algorithm is designed so that it never fails even if numerical computation contains large errors. To see this property, artificial errors were added to all the floating-point computations in the algorithm using random numbers. Then, the output becomes as shown in Fig. 5(b). Some part of the output is not correct. However, what is important is that although the algorithm made misjudgements, it carries out the task, ending up with some output. When we added larger artificial errors, the output becomes as shown in Fig. 5(c). As the extremal case, when we replaced all the floating-point computations by random numbers, then the output was as shown in Fig. 5. This output is of course nonsense, but an important thing is that topological inconsistency never arises in this algorithm and always some output is given. If we see Fig. 5(d), (c), (b), (a) in this order, we can say that the output of

the algorithm converges to the correct answer as the precision in computation becomes higher.

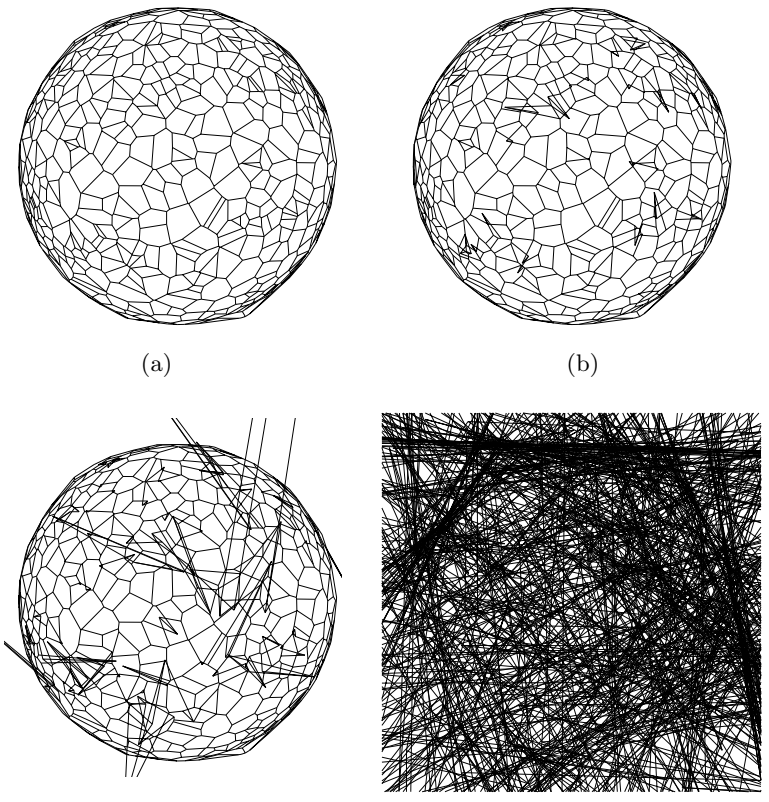
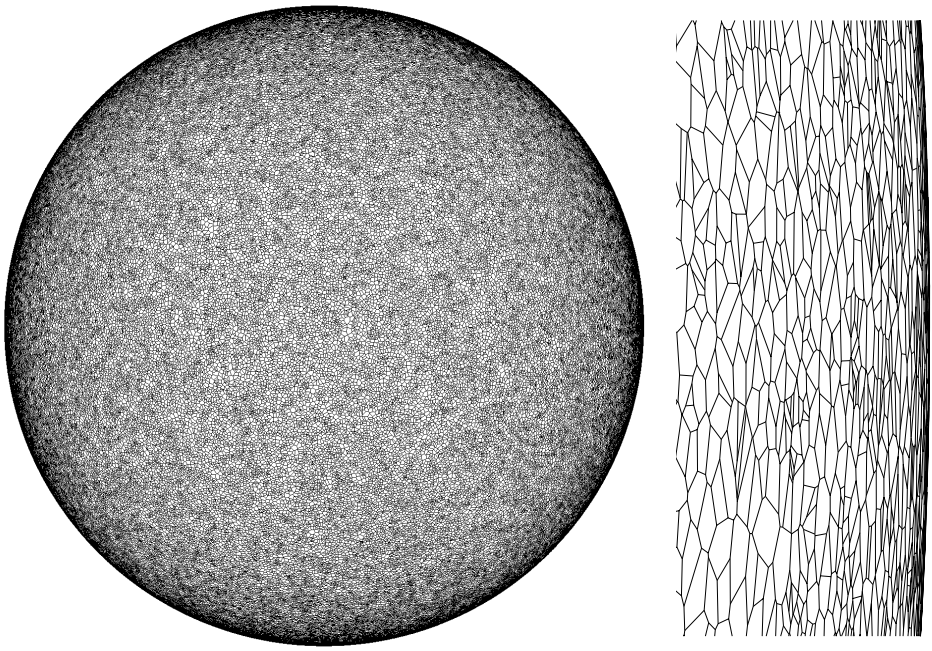


Fig. 5. Behavior of the topology-oriented algorithm for cutting a polyhedron by a plane.

Another example of the output of this algorithm is shown in Fig. 6. Fig. 6(a) is the result of cutting a cube by 10^5 planes touching a common sphere, and Fig. 6(b) is a magnified picture of the left portion. This example also shows the robustness of the algorithm.

5.2 Construction of Voronoi Diagrams

Let $S = \{\mathrm{P}_1, \mathrm{P}_2, \ldots, \mathrm{P}_n\}$ be a set of finite number of points in the plane. The region $R(S; \mathrm{P}_j)$ defined by $R(S; \mathrm{P}_i) = \{\mathrm{P} \in \mathbf{R}^2 \mid d(\mathrm{P}, \mathrm{P}_i) < d(\mathrm{P}, \mathrm{P}_j),\ j =$



(a) (b)

Fig. 6. Cutting a cube by 10^5 planes.

$1, \ldots, i-1, i+1, \ldots, n\}$ is called the *Voronoi region* of P_i, where $d(\mathrm{P}, \mathrm{Q})$ represents the Euclidean distance between the two points P and Q. The partition of the plane into Voronoi region $R(S; \mathrm{P}_i)$, $i = 1, 2, \ldots, n$, and their boundaries is called the *Voronoi diagram* for S.

In the incremental algorithm, we start with the Voronoi diagram for a few points, and modify it by adding the other points one by one. An increment step proceeds in the following way. Suppose that we have already constructed the Voronoi diagram for k points, and now want to add the $(k+1)$-th point. To modify the Voronoi diagram, we first find a cyclic list formed by the perpendicular bisectors between the new point and the neighboring old points, and next remove the substructure inside this cycle.

Though this procedure is theoretically simple, it is numerically unstable because the sequence of bisectors does not necessarily form a cycle in imprecise arithmetic, particularly when the input points are degenerated [42,43]. To construct a robust algorithm, we can use the following property.

Proposition 2. If a new point is inside the convex hull of the old points, the substructure to be removed is a tree in a graph theoretical sense.

We place higher priority on this property than on numerical values, and thus can construct a numerically robust algorithm for the Voronoi diagram [42,43]. Fig. 7(a) is an example of the output of this algorithm. Though the points were highly degenerate, the algorithm constructed the globally correct Voronoi diagram. If we magnify the central portion of this figure by 10^4, 10^5 and 10^6 respectively, we can see small disturbance, as shown in Fig. 7(b), (c) and (d). However, it should be noted that such disturbance never makes the algorithm to clash, because the algorithm always maintains topological consistency of the data structure.

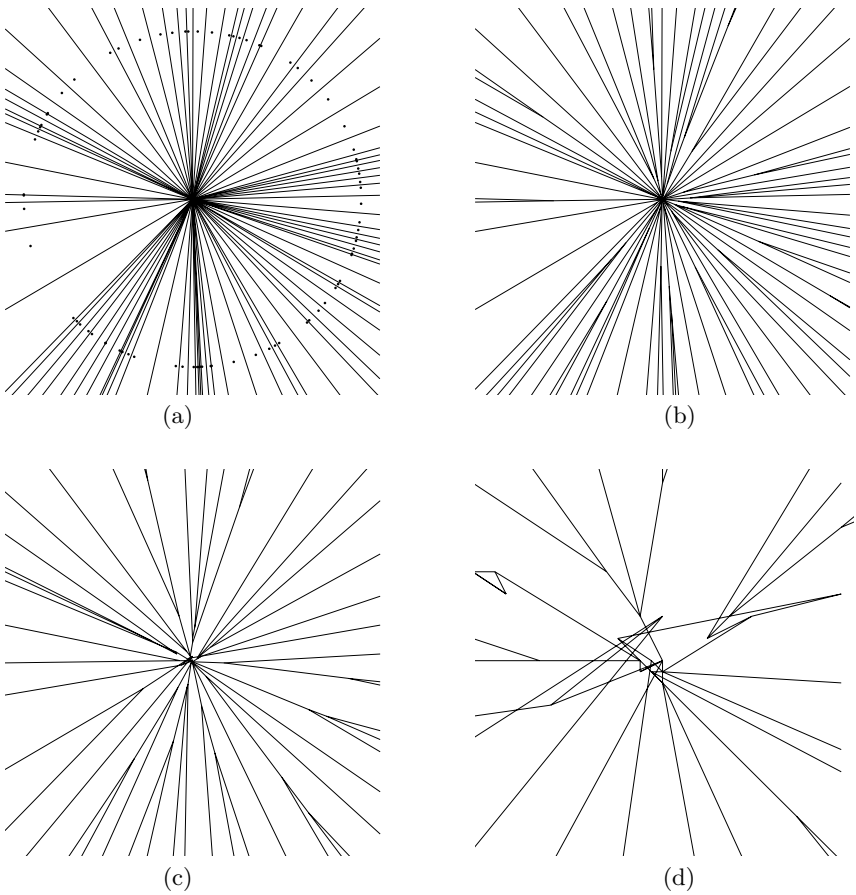


Fig. 7. Voronoi diagram for highly degenerate set of points

Other applications of the topology-oriented method include the divide-and-conquer construction of the two-dimensional Voronoi and Delaunay diagrams [28], the incremental construction of the three-dimensional Voronoi and Delau-

nay diagrams [20,19], the incremental construction of the Voronoi diagram for polygons [17], the gift-wrapping construction of the three-dimensional convex hull [39], the divide-and-conquer construction of the three-dimensional convex hull [26,27], the intersection of half spaces in the three-dimensional space [38], and other applications [36,37].

6 Discussions

Here we consider some general properties of the topology-oriented algorithms.

Robustness

A topology-oriented algorithm is completely robust in the sense that it does not require any minimum precision in numerical computation. All possible behavior is specified by the topological skeleton, and therefore even if numerical precision is very poor (or even if all the results of numerical computation are replaced by random numbers), the algorithm still carries out the task and generates some output.

Topological Consistency

Whether the algorithm is topologically consistent depends on the chosen set Q of purely topological properties. The topology-oriented implementation guarantees that the output satisfies all the properties in Q. In general, however, Q gives only a necessary condition for the output to belong to the set $\Omega(P)$ of all the possible solutions of the problem P; it does not necessarily give a sufficient condition. This is because the purely topological characterization of the solution set is not known for many geometric problems, and even if it is known, it is usually time-consuming to check the conditions (note that Q should contain only those properties that can be checked efficiently).

Hence, topological consistency can be attained for a limited number of problems. A trivial example is the problem of constructing a convex hull in the plane. For this problem, any cyclic sequence of three or more vertices chosen from the input points can be the solution of a perturbed version of the input, so that topological consistency can be easily attained.

More nontrivial examples arise in the class of problems related to convex polyhedra. The topological structures of convex polyhedra can be characterized by Steinitz's theorem, which says that graph G is a vertex-edge graph of a convex polyhedron if and only if G is a 3-connected planar graph with four or more vertices [33]. Because of this theorem we can see that the algorithm in Section 5.1 is topologically consistent. Actually we can prove that if the input graph G is a 3-connected planar graph, then the output G' is also a 3-connected planar graph. Hence, the output of this algorithm is the vertex-edge graph of some polyhedron, that is, the output is the vertex-edge graph of the solution of some instance of the problem though it is not necessarily the given instance.

For the two-dimensional Voronoi diagram for points, necessary and sufficient conditions are known [13,14]. However, these conditions require much time to

check, and hence cannot be included in Q. Actually the algorithm in Section 5.2 uses only a necessary condition, and hence it is not topologically consistent.

Convergence

If the input to the algorithm is not degenerate, the output converges to the correct solution as the computation becomes more and more precise, because the correct branch of the processing is chosen with sufficiently high precision. However, the speed of convergence cannot be stated in a unifying manner, because it depends on the individual problems and on the implementation of numerical computation.

The situation is different for degenerate input. If the algorithm is topologically consistent, the output converges to an infinitesimally perturbed version of the correct solution. In any high precision, the true degenerate output cannot be obtained, because degenerate cases are not taken into account in the topology-oriented approach. For example, suppose that the cutting plane H goes through a vertex of the polyhedron Π. Then our algorithm classifies the vertex either above H or below H, and decides the topological structure accordingly. As a result, the output may contain the edges whose lengths are almost 0.

7 Concluding Remarks and Open Problems

We have seen the topology-oriented approach to the robust implementation of geometric algorithms, and also discussed related issues. Since we can separate the topological-inconsistency issue from the error-analysis issue completely, the algorithm designed in this approach has the following advantages:

(1) No matter how large the numerical errors are that may take place, the algorithm never fails; it always carries out the task and gives some output.

(2) The output is guaranteed to satisfy the topological properties Q used in the topological skeleton of the algorithm.

(3) For a nondegenerate input, the output converges to the correct solution as the precision in computation becomes higher.

(4) The structure of the algorithm is simple because exceptional branches for a degenerate input are not necessary.

However, in order to use the output for practical applications we still have many problems to be solved.

The topology-oriented approach might give output that contains numerical disturbance particularly when the input is close to degeneracy. Such disturbances are usually very small but not acceptable for some applications. Hence, to rewrite the application algorithms in such a way that they can use numerically disturbed output of the topology-oriented algorithms is one of main future problems related to this approach.

This work is supported by the Grant-in-Aid for Scientific Research of the Japan Ministry of Education, Science, Sports, and Culture, and the Torey Science Foundation.

References

1. M. Benouamer, D. Michelucci and B. Peroche: Error-free boundary evaluation using lazy rational arithmetic—A detailed implementation. *Proceedings of the 2nd Symposium on Solid Modeling and Applications*, Montreal, 1993, pp. 115–126.
2. H. Brönnimann, I. Z. Emiris, V. Y. Pan and S. Pion: Computing exact geometric predicates using modular arithmetic with single precision. *Proceedings of the 13th Annual ACM Symposium on Computational Geometry*, Nice, June 1997, pp. 1-182.
3. H. Brönnimann and M. Yvinec: Efficient exact evaluation of signs of determinants. *Proceedings of the 13th Annual ACM Symposium on Computational Geometry*, Nice, June 1997, pp. 166-173.
4. K. L. Clarkson: Safe and effective determinant evaluation. *Proceedings of the 33rd IEEE Symposium on Foundation of Computer Science*, pp. 387-395.
5. D. Dobkin and D. Silver: Recipes for geometric and numerical analysis—Part I, An empirical study. *Proceedings of the 4th ACM Annual Symposium on Computational Geometry*, Urbana-Champaign, 1988, pp. 93–105.
6. H. Edelsbrunner and E. P. Mücke: Simulation of simplicity—A technique to cope with degenerate cases in geometric algorithms. *Proceedings of the 4th ACM Annual Symposium on Computational Geometry*, Urbana-Champaign, 1988, pp. 118–133.
7. D. A. Field: Mathematical problems in solid modeling—A brief survey. G. E. Farin (ed.), *Geometric Modeling—Algorithms and New Trends*, SIAM, Philadelphia, 1987, pp. 91–107.
8. S. Fortune: Stable maintenance of point-set triangulations in two dimensions. *Proceedings of the 30th IEEE Annual Symposium on Foundations of Computer Science*, Research Triangle Park, California, 1989, pp.94–499.
9. S. Fortune: Numerical stability of algorithms for 2D Delaunay triangulations. *International Journal of Computational Geometry and Applications*, vol. 5 (1995), pp. 193-213.
10. S. Fortune and C. von Wyk: Efficient exact arithmetic for computational geometry. *Proceedings of the 9th ACM Annual Symposium on Computational Geometry*, San Diego, 1993, pp. 163–172.
11. D. H. Greene and F. Yao: Finite resolution computational geometry. *Proceedings of the 27th IEEE Symposium on Foundations of Computer Science*, Toronto, October 1986, pp. 3-152.
12. L. Guibas, D. Salesin and J. Stolfi: Epsilon geometry—Building robust algorithms from imprecise computations. *Proc. 5th ACM Annual Symposium on Computational Geometry* (Saarbrücken, May 1989), pp. 208–217.
13. T. Hiroshima, Y. Miyamoto and K. Sugihara: Another proof of polynomial-time recognizability of Delaunay graphs. *IEICE Transactions on Fundamentals*, Vol. E83-A (2000), pp. 627-638.
14. C. D. Hodgson, I. Rivin and W. D. Smith: A characterization of convex hyperbolic polyhedra and of convex polyhedra inscribed in the sphere. *Bulletin of the American Mathematical Society*, vol. 27 (1992), pp. 6-251.
15. C. M. Hoffmann: The problems of accuracy and robustness in geometric computation. *IEEE Computer*, vol. 22, no. 3 (March 1989), pp. 31-1.

16. C. M. Hoffmann: *Geometric and Solid Modeling*. Morgan Kaufmann Publisher, San Mateo, 1989.
17. T. Imai: A topology-oriented algorithm for the Voronoi diagram of polygon. *Proceedings of the 8th Canadian Conference on Computational Geometry*, 1996, pp. 107–112.
18. T. Imai: How to get the sign of integers from their residuals. *Abstracts of the 9th Franco-Japan Days on Combinatorics and Optimization*, 1996, p. 7.
19. H. Inagaki and K. Sugihara: Numerically robust algorithm for constructing constrained Delaunay triangulation. *Proceedings of the 6th Canadian Conference on Computational Geometry*, Saskatoon, August 19, pp. 171-176.
20. H. Inagaki, K. Sugihara and N. Sugie, N.: Numerically robust incremental algorithm for constructing three-dimensional Voronoi diagrams. *Proceedings of the 6th Canadian Conference Computational Geometry*, Newfoundland, August 1992, pp. 3–339.
21. M. Karasick, D. Lieber and L. R. Nackman: Efficient Delaunay triangulation using rational arithmetic. *ACM Transactions on Graphics*, vol. 10 (1991), pp. 71–91.
22. D. E. Knuth: *Axioms and Hulls*. Lecture Notes in Computer Science, no. 606, Springer-Verlag, Berlin, 1992.
23. G. Liotta, F. P. Preparata and R. Tamassia: Robust proximity queries — An illustration of degree-driven algorithm design. *Proceedings of the 13th Annual ACM Symposium on Computational Geometry*, 1997, pp. 156-165.
24. K. Mehlhorn and S. Näher: A platform for combinatorial and geometric computing. *Communications of the ACM*, January 1995, pp. 96-102.
25. V. Milenkovic: Verifiable implementations of geometric algorithms using finite precision arithmetic. *Artificial Intelligence*, vol. 37 (1988), pp. 377-01.
26. T. Minakawa and K. Sugihara: Topology oriented vs. exact arithmetic—experience in implementing the three-dimensional convex hull algorithm. H. W. Leong, H. Imai and S. Jain (eds.): *Algorithms and Computation, 8th International Symposium, ISAAC'97* (Lecture Notes in Computer Science 1350), (December, 1997, Singapore), pp. 273–282.
27. T. Minakawa and K. Sugihara: Topology-oriented construction of three-dimensional convex hulls. *Optimization Methods and Software*, vol. 10 (1998), pp. 357–371.
28. Y. Oishi and K. Sugihara: Topology-oriented divide-and-conquer algorithm for Voronoi diagrams. *Computer Vision, Graphics, and Image Processing: Graphical Models and Image Processing*, vol. 57 (1995), pp. 303–3.
29. T. Ottmann, G. Thiemt and C. Ullrich: Numerical stability of geometric algorithms. *Proceedings of the 3rd ACM Annual Symposium on Computational Geometry*, Waterloo, 1987, pp. 119–125.
30. P. Schorn: Robust algorithms in a program library for geometric computation. Dissertation submitted to the Swiss Federal Institute of Technology (ETH) Zürich for the degree of Doctor of Technical Sciences, 1991.
31. M. Segal and C. H. Sequin: Consistent calculations for solid modeling. *Proceedings of the ACM Annual Symposium on Computational Geometry*, Baltimore, 1985, pp. 29–38.
32. J. R. Shewchuk: Robust adaptive floating-point geometric predicates. *Proceedings of the 12th Annual ACM Symposium on Computational Geometry*, Philadelphia, May 1996, pp. 1-150.
33. E. Steinitz: *Polyheder und Raumeinteilungen*. Encyklopädie der mathematischen Wissenchaften, Band III, Teil 1, 2. Hälfte, IIIAB12, pp. 1-139.

34. A. J. Steward: Local robustness and its application to polyhedral intersection. *International Journal of Computational Geometry and Applications*, vol. (1994), pp. 87-118.
35. K. Sugihara: A simple method for avoiding numerical errors and degeneracy in Voronoi diagram construction. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, vol. E75-A (1992), pp.68–477.
36. K. Sugihara: An intersection algorithm based on Delaunay triangulation. *IEEE Computer Graphics and Applications*, vol. 12, no. 2 (March 1992), pp. 59-67.
37. K. Sugihara: Approximation of generalized Voronoi diagrams by ordinary Voronoi diagrams. *Computer Vision, Graphics, and Image Processing: Graphical Models and Image Processing*, vol. 55 (1993), pp. 522-531.
38. K. Sugihara: A robust and consistent algorithm for intersecting convex polyhedra. *Computer Graphics Forum*, EUROGRAPHICS'94, Oslo, 1994, pp. C-45–C-54.
39. K. Sugihara: Robust gift wrapping for the three-dimensional convex hull. *Journal of Computer and System Sciences*, vol.9 (1994), pp. 391–407.
40. K. Sugihara: Experimental study on acceleration of an exact-arithmetic geometric algorithm. *Proceedings of the 1997 International Conference on Shape Modeling and Applications*, Aizu-Wakamatsu, 1997, pp. 160–168.
41. K. Sugihara and M. Iri: A solid modelling system free from topological inconsistency. *Journal of Information Processing*, vol. 12 (1989), pp. 380–393.
42. K. Sugihara and M. Iri: Construction of the Voronoi diagram for "one million" generators in single-precision arithmetic. *Proceedings of the IEEE*, vol. 80 (1992), pp. 71–1484.
43. K. Sugihara and M. Iri: A robust topology-oriented incremental algorithm for Voronoi diagrams. *International Journal of Computational Geometry and Applications*, vol. (1994), pp. 179–228.
44. C. K. Yap: A geometric consistency theorem for a symbolic perturbation scheme. *Proceedings of the 4th Annual ACM Symposium on Computational Geometry*, Urbana-Champaign, 1988, pp. 1–142.
45. C. K. Yap: The exact computation paradigm. D.-Z. Du and F. Hwang (eds.): *Computing in Euclidean Geometry, 2nd edition*. World Scientific, Singapore, 1995, pp.52–492.
46. X. Zhu, S. Fang and B. D. Brüderlin: Obtaining robust Boolean set operations for manifold solids by avoiding and eliminating redundancy. *Proceedings of the 2nd Symposium on Solid Modeling and Applications*, Montreal, May 1993, pp. 7-154.

Metacomputing with the Harness and IceT Systems

Vaidy Sunderam

Dept of Math. and Computer Science
Emory University
1784 N. Decatur Rd.
Atlanta, GA 30322, USA
Phone: +1-(404)-727-5926
FAX: +1-(404)-727-5611
vss@mathcs.emory.edu

Abstract

Metacomputing, or network-based concurrent processing, has evolved over the past decade from an experimental methodology to a mainstream technology. We use the term metacomputing in a broad sense to include clusters of workstations with high-speed interconnects, loosely coupled local network clusters, and wide area configurations spanning multiple architectures, machine ranges, and administrative domains. These modes of distributed computing are proving to be highly viable platforms for a wide range of applications, primarily in the high-performance scientific computing domain, but also in other areas, notably web search engines and large databases. From the systems point of view, metacomputing technologies are being driven primarily by new network and switch technologies in closely coupled systems, and by software advances in protocols, tools, and novel runtime paradigms. This short course will discuss two alternative approaches to metacomputing that the Harness and IceT projects are investigating.

Harness is a metacomputing framework based on dynamic reconfigurability and extensible distributed virtual machines. The Harness system seeks to achieve two important goals. First, by enabling reconfiguration of the facilities provided by the virtual machine, Harness is able to provide specialized services appropriate to the platform and adapt to new technological developments. Second, Harness is able to adapt to application needs by configuring the required support services and programming environments on demand. In this talk, we describe the architecture and key features of Harness, and discuss preliminary experiences with its use. IceT is a system being developed to support collaborative metacomputing. While the focus of Harness is on reconfigurability, IceT is aimed at sharing of resources by merging and splitting virtual machines. Multiple users owning different sets of resources may occasionally pool them as problem situations dictate; IceT provides a structured framework and context for this type of sharing, and addresses security and resource management issues. An overview of the IceT system, and a discussion of its salient features will be presented in this talk.

V.N. Alexandrov et al. (Eds.): ICCS 2001, LNCS 2073, p. 27, 2001.
© Springer-Verlag Berlin Heidelberg 2001

Computational Biology: IT Challenges and Opportunities

Stefan Unger, PhD*, and Andrew Komornicki, PhD

Sun Microsystems
Menlo Park, CA
USA
Phone: 1-650-786-0310 (80310)
{Stefan.Unger|Andrew.Komornicki}@eng.sun.com

Abstract

We will survey the field of computational biology and discuss the many interesting computational challenges and opportunities in areas such as genomics, functional and structural genomics, pharmacogenomics, combinatorial chemistry/high throughput screening, and others of current interest.

V.N. Alexandrov et al. (Eds.): ICCS 2001, LNCS 2073, p. 28, 2001.
© Springer-Verlag Berlin Heidelberg 2001

A Data Broker for Distributed Computing Environments

L.A. Drummond[1], J. Demmel[3], C.R. Mechoso[2], H. Robinson[3], K. Sklower[3], and J.A. Spahr[2]

[1] National Energy Research Scientific Computing Center, Lawrence Berkeley National Laboratory
Berkeley, CA 94720, USA
Ladrummond@lbl.gov
[2] Department of Atmospheric Sciences, University of California, Los Angeles,
Los Angeles, CA 90095-1565, USA
{mechoso,spahr}@atmos.ucla.edu
[3] Computer Science Division, University of California, Berkeley
Berkeley, CA 94720-1776, USA
{demmel,hbr,sklower}@cs.berkeley.edu

Abstract. This paper presents a toolkit for managing distributed communication in multi-application systems that are targeted to run in high performance computing environments; the Distributed Data Broker (DDB). The DDB provides a flexible mechanism for coupling codes with different grid resolutions and data representations. The target applications are coupled systems that deal with large volumes of data exchanges and/or are computational expensive. These application codes need to run efficiently in massively parallel computer environments generating a need for a distributed coupling to minimize long synchronization points. Furthermore, with the DDB, coupling is realized in a plug-in manner rather than hard-wire inclusion of any programming language statements. The DDB performance in the CRAY T3E-600 and T3E-900 systems is examined

Keywords: MMP systems, Distributed Computing, Data Brokerage, coupling.

1 Introduction

The Distributed Data Broker (DDB) is a toolkit for managing distributed communication in multi-application systems that run coupled in high performance computing environments. The DDB evolved from a Data Broker designed as a part of a coupled atmosphere-ocean modeling system, in which the model components can work on different horizontal resolutions, grid representations and cover different geographical domains [1]. The high efficiency demanded by those codes in massively parallel computer environments generated a need for extending the Data Broker in a way that minimizes long synchronization points inside model components and memory bottlenecks. Using the DDB, applications are integrated to the coupled system in a plug-in manner rather than by hard-wire inclusion of any programming language statements. The DDB was designed under a consumer-producer paradigm,

V.N. Alexandrov et al. (Eds.): ICCS 2001, LNCS 2073, pp. 31-40, 2001.
© Springer-Verlag Berlin Heidelberg 2001

in which, an application produces data to be consumed by one or more applications, and an application can be a consumer, producer or both.
This paper is an introduction to the DDB tool. Section 2 presents a summary of the functionality of the DDB and its library components. A general example of a coupled application using the tool is described in Section 3. Performance results are shown in Section 4.

2 The Distributed Data Broker

The functionality of the DDB is encapsulated in a modular design that contains three libraries of routines that are built to work together and called on demand from different places inside the codes to be coupled. These DDB components are the Communication Library (CL), The Model Communication Library (MCL) and the Data Translation Library (DTL). The CL is the core library of routines that it is used to implement the point-to-point communication between computational nodes in a distributed environment. This DDB component encapsulates the functionality of widely used message passing software like PVM 3 or MPI into the DDB context. A more technical description of the CL is presented in [3]

Two types of steps characterize the coupling of applications using the DDB; an initial registration step and subsequent data communication steps. The MCL provides an API (Application Programming Interface) that supports the implementation of both steps from C or Fortran programs. The registration step is an initial code "handshake" in which different codes exchange information about the production and consumption of data. The registration step begins with the identification of a process or a task as the Registration Broker (RB) with a call to the `MCLIamRegisgtrationBroker` routine. There is only one RB per coupled run and this task only exists during the registration step. This implies that after the registration step, the process acting as the RB can perform any other tasks inside one of the applications being coupled. In addition, each application must identified a control process (CP), each CP is responsible for reporting global domain information to the application like grid resolution, number of processes, data layout, and frequency of production or consumption to the RB. The RB can also be the CP for the application that spawns it.

The RB starts collecting information from all the CP's. Then, the RB processes this information to match producers against consumers. For example if Process 1 of Model A is designated as the RB as in Fig. 1, this collects general information from process 1 of Model B (such as global domain, grid resolution, number of processes, and frequencies of production and consumption. Without loss of generality, in this example we depicted one model that works in a wider domain than the other, and uses a different horizontal grid spacing. The DDB will also works with equal domains, and equal grid spacing as long as a geographical domain can be mapped into the other domain and a grid translation function exist between both grids.

Lastly, the registration step ends with a call to `MCLRegistration` from all other processes participating in the coupling to register their process id and subdomain. Every process receives back a list of processes that it will exchange data with at execution time. As a result, every participating process in the coupling has enough information to send and receive data from its peers without the need of a centralized entity regulating the exchanges of information.

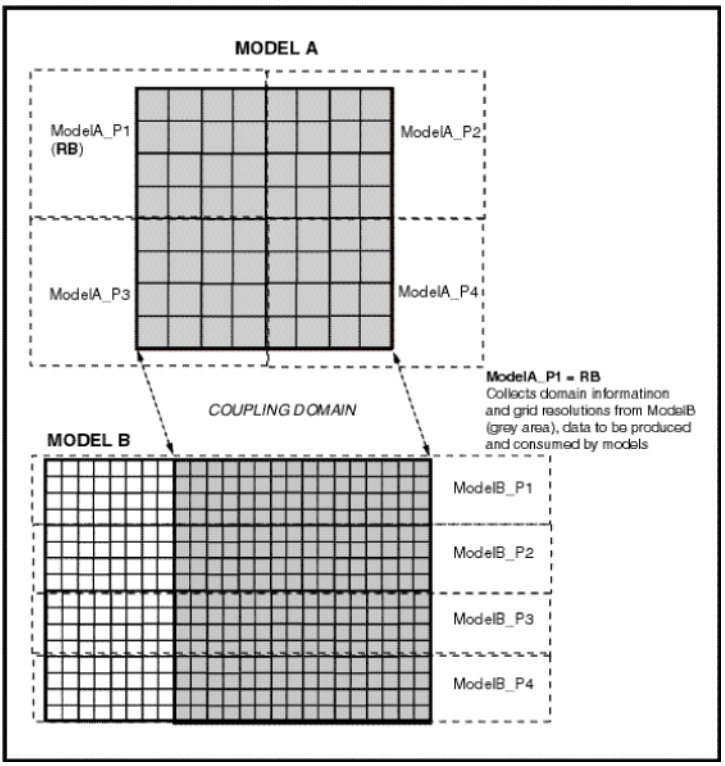


Fig. 1. DDB registration step. Registration Broker (*RB*) collects information of every model (i.e., *model A* and *model B*). This information includes: model's resolution, domain, offers for data production, requests for data production, frequency of consumption and production of data, and parallel data layout.

Fig. 2 presents an schematic of all the MCL routines that implement the registration and communication steps. The communication step is characterized by patterns of communication between the coupled components. A producer code that wants to send data to its consumers, will simply execute a call to `MCLSendData`, which gets translated into several CL commands that in turn call the MPI or PVM libraries to complete the communication step. Thus, the MCL-CL interface provides a level of transparency and code portability because the communication syntax used inside a program remains invariant when porting the code from PVM 3 to MPI or vice-versa and these communication packages in turn provide portability across platforms.

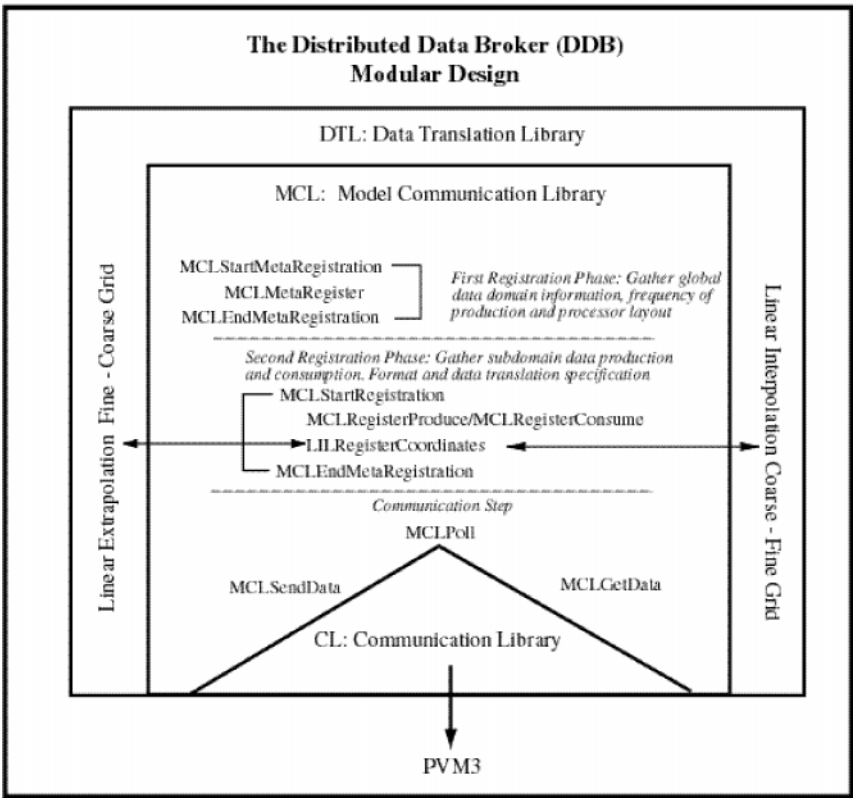


Fig. 2. Schematic of the DDB. The Application Programming Interface is provided via the MCL. In turn the MCL makes use of the CL library to interface with standard message passing libraries like PVM and the user-defined Data Translation Libraries. The current DDB has implemented a Linear Interpolation Library (LIL) of routines

The basic MCL communication phase has two operations, `MCLGetData` and `MCLSendData`. A user's call to `MCLSendData` automatically generates one or many calls to the send-routine in the CL library, one per consumer of the data produced (e.g., one `pvmfsend` per consumer). Similarly, a user's call to `MCLGetData` automatically receives one or many messages, pastes them together and transforms them into compatible data for the consumer's grid using a predefined DTL routine. The DTL component handles the data transformations from a producer's grid to the consumer's grid. The DTL routines are invoked by certain calls to the MCL that deliver data at the consumer end (i.e., `MCLGetData`) The DTL can include several numerical transformation routines and the user can decided the transformation algorithm to be used according to the numerical requirements f the applications. In any case, the calls to the MCL library remain the same but each of the low-level transformation routines in the DTL are overloaded with different

procedures depending on the context. In view our current coupling scenarios and requirements for data transformations, we have implemented a set of linear interpolation routines.

3 An Example of Coupling with the DDB

The current version of the Distributed Data Broker (DDB) is being used to couple different model components of the UCLA Earth System Model (ESM) under the NASA/ESS HPPC program. In this system the model components are parallel codes that in turn run in parallel exchanging atmospheric or oceanic fields in a prescribed time intervals. In conventional couplers, these data exchanges and translations are handled using a centralized global domain algorithm. Here we present a fully distributed approach to coupling in which the data translations between models are handled in parallel and using a subdomain based numerical algorithms. The DDB approach to coupling promotes high levels computational efficiency by reducing the number of synchronization points, the need of global reductions operations, and idle nodes in the system.

The UCLA Atmospheric General Circulation Model (AGCM) is a state of the art grid point model of the global atmosphere ([2],[5]) extending from the Earth's surface to a height of 50 km. The model predicts the horizontal wind, potential temperature, water vapor mixing ratio, planetary boundary layer (PBL) depth and the surface pressure, as well as the surface temperature and snow depth over land. The Oceanic General Circulation Model is the Parallel Ocean Program (POP), which is also based on a two-dimensional (longitude-latitude) domain decomposition [4], and uses message passing to handle data exchanges between distributed processors.

The UCLA AGCM is a complex code representing many physical processes. Despite the complexity of the code, one can identify the following two major components:

- **AGCM/Dynamics**, which computes the evolution of the fluid flow governed by the appropriate equations (the primitive equations) written in finite differences.
- **AGCM/Physics**, which computes the effect of processes not resolved by the model's grid (such as convection on cloud scales) on processes that are resolved by the grid (such as the flow on the large scale).

The OGCM also has two major components:

- **OGCM/Baroclinic**, determines the deviation from the vertically averaged velocity, temperature and salinity fields.
- **OGCM/Barotropic**, determines the vertically averaged distributions of those fields.

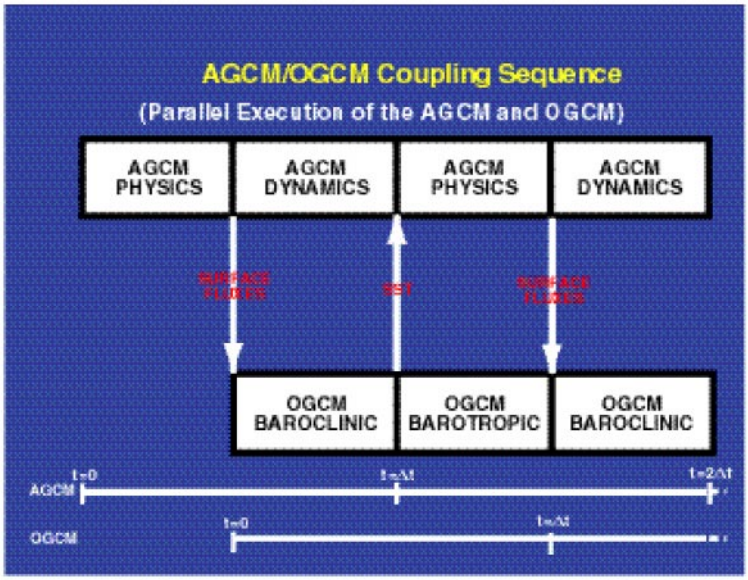


Fig. 3. Distributed AGCM-OGCM coupling. The AGCM send surface fluxes to the OGCM and receives in return Sea Surface Temperature. This exchanges happen at regular intervals t

The coupled atmosphere-ocean GCM, therefore, can be decomposed into four components. When run on a single node the AGCM and OGCM codes execute sequentially and exchange information corresponding to the air-sea interface. The AGCM is first integrated for a fixed period of time and then transfers the time-averaged surface wind stress, heat and water fluxes to the OGCM. This component is then integrated for the same period of time and transfers the sea surface temperature to the AGCM. The data transfers, including the interpolations required by differences in grid resolution between model components, was originally performed by a suite of coupling routines and we refer to this approach as the centralize coupling approach. Coupling with the DDB is realized with a registration step followed by model computations and inter-model communication handled by MCLGetData and MCLSendData calls. The necessary data translations are also performed under these calls.

The coupled GCM runs in a parallel environment following the scheme depicted in Fig. 3, which allows the two codes to run in parallel. Because there are no data dependencies between the AGCM/Dynamics and the OGCM/Baroclinic, these components can run in parallel. Further, AGCM/Physics can start as soon as OGCM/Baroclinic completes its calculation, because this module provides the sea surface temperature. Similarly, The AGCM/Physics can run in parallel with OGCM/Barotropic.

4 Performance Results

This section presents some results obtained from running the coupled UCLA AGCM/OGCM model described in section 3. We compare here the centralized coupling against the decentralized one. Fig. 4 to Fig. 6 shows the model resolutions used in each case, and compare the memory and time required by the coupling interfaces

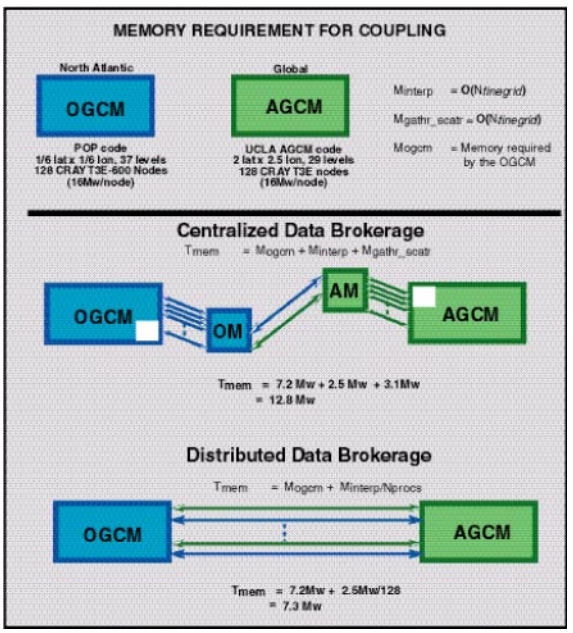


Fig. 4. Memory requirements for centralized and distributed coupling.

Fig. 4 and Fig. 5 illustrate comparison results based on the memory requirements for both coupling implementations. In Fig. 4, the centralized data brokerage requires almost twice as much memory as the distributed data brokerage because it needs to collect the entire grid from one model in a single node. In the distributed case, each processor has enough information to produce the data needed by consumer processes and communication is realized in distributed manner. In Fig. 5, a more drastic scenario is presented, in which the centralized coupling cannot be realized because of the 45Mw memory requested in a single computational node. In this case the distributed case requires less than a third of the memory requested by the centralized approach.

Fig. 6 compares the execution time between the two coupling approaches, and in this case the AGCM is sending 4 fields to the OGCM, and the requested time by the distributed approach is one third of the centralized. In the reverse communication, the

OGCM sends a single field to the AGCM and the requested time is also greatly reduced with the distributed approach.

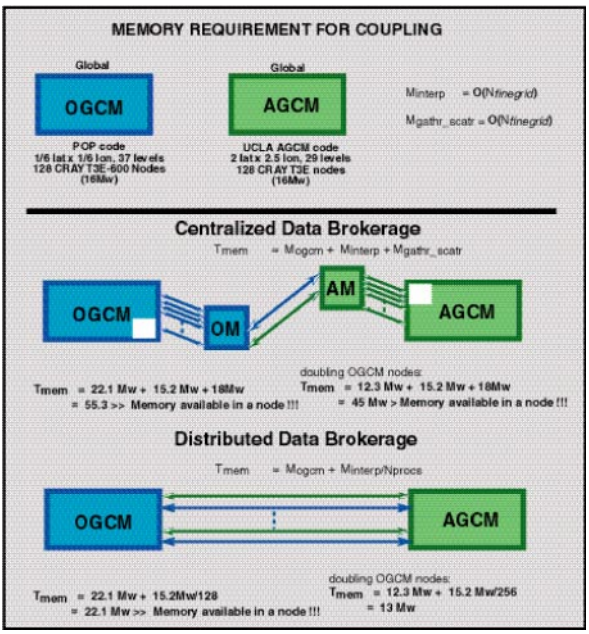


Fig. 5. Memory requirements for centralized and distributed coupling. First, we double the resolution OGCM resolution and increase the number of nodes

Fig. 7 presents the asymptotic behavior of centralized vs. distributed coupling. As indicated the number of seconds required by coupling the AGCM/OGCM in the centralized case (one process case) grows exponentially as the problem size is increased. The time requested by the distributed coupling approach, the DDB, is reduced as the number of processes is also increased.

5 Conclusions

As computational sciences continue to push forward the frontier of knowledge about physical phenomena, more complex models are and will be developed to enable their computerized simulations. The demand for computational resources to carry out these simulations will also increase, as well as the need for optimized tools that help application developers to make better of use of the available resources. The DDB addresses not only the issues of optimal coupling, but also provides a flexible approach to coupling models and applications in a "plug-and-play" manner rather than intrusive coding in the applications.

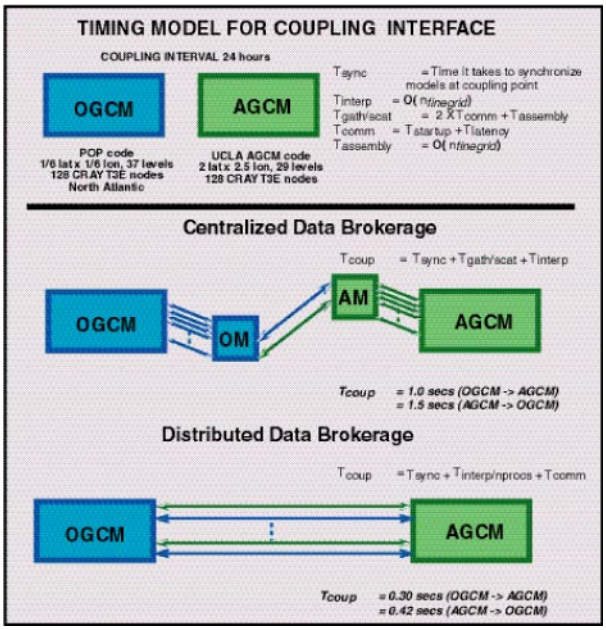


Fig. 6. Simplified Timing model of centralized vs. distributed coupling.

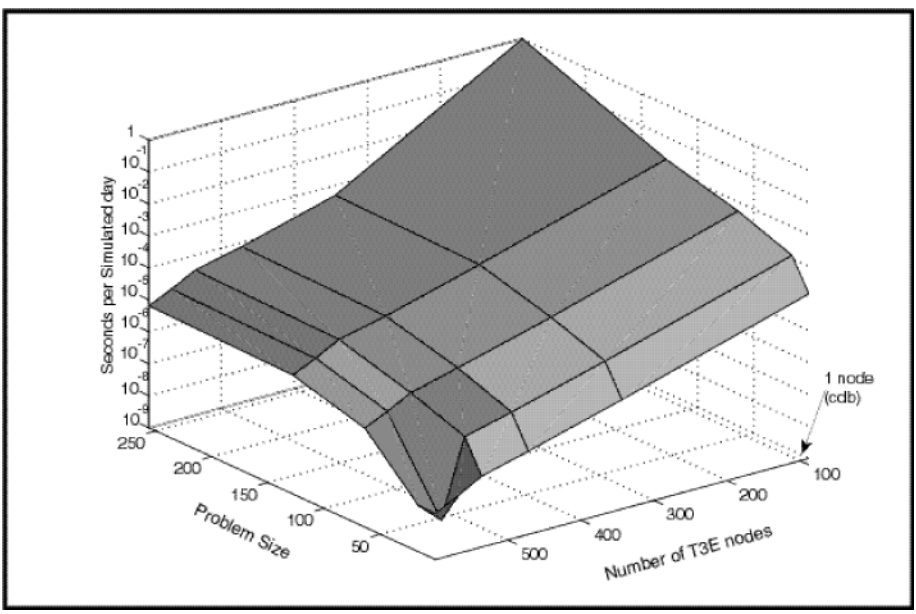


Fig. 7. Asymptotic behavior of centralized vs. distributed coupling .

Further development of the DDB is still under way at University of California, Los Angeles and collaborators at UC Berkeley. Future agenda includes the inclusion of higher order interpolations for data translations, use of other communication libraries such as MPI, and continue to prototype other scientific applications using the DDB technology.

Acknowledgements

This project has been supported by the NASA High Performance Computing and Communication for Earth and Space Sciences (HPCC-ESS) project under CAN 21425/041. The tests were performed at the Department of Energy's National Energy Research Scientific Computing center (NERSC)

References

1. Drummond, L. A., J. D. Farrara, C. R. Mechoso, J. A. Spahr, J. W. Demmel, K. Sklower and H. Robinson, 1999: An Earth System Model for MPP environments: Issues in coupling components with different complexities. *Proceedings of the 1999 High Performance Computing - Grand Challenges in Computer Simulation Conference* April 11-15, 1999, San Diego, CA, 123-127.
2. Mechoso, C. R., L. A. Drummond, J. D. Farrara, J. A. Spahr, 1998: The UCLA AGCM in high performance computing environments. *In Proceedings, Supercomputing 98*, Orlando, FL.
3. Sklower, K., H.R. Robinson, L.A. Drummond, C.R. Mechoso, J. A. Spahr, E. Mesrobian, 2000: The Data Broker: A decentralized mechanism for periodic exchange of fields between multiple ensembles of parallel computations http://www.cs.berkeley.edu/~sklower/DDB/paper.html
4. Smith, R.D., J.K. Dukowicz, and R.C. Malone, 1992: Parallel Ocean General Circulation Modeling, *Physica D*, 60, 38-61.
5. Wehner, M. F., A. A. Mirin, P. G. Eltgroth, W. P. Dannevik, C. R. Mechoso, J. D. Farrara and J. A. Spahr, 1995: Performance of a distributed memory finite-difference atmospheric general circulation model. *Parallel Computing*, 21, 1655-1675.

Towards an Accurate Model for Collective Communications*

Sathish S. Vadhiyar, Graham E. Fagg, and Jack J. Dongarra

Computer Science Department
University of Tennessee, Knoxville
{vss, fagg, dongarra}@cs.utk.edu

Abstract. The performance of the MPI's collective communications is critical in most MPI-based applications. A general algorithm for a given collective communication operation may not give good performance on all systems due to the differences in architectures, network parameters and the storage capacity of the underlying MPI implementation. Hence, collective communications have to be tuned for the system on which they will be executed. In order to determine the optimum parameters of collective communications on a given system in a time-efficient manner, the collective communications need to be modeled efficiently. In this paper, we discuss various techniques for modeling collective communications .

1 Introduction

This project developed out of an attempt to build efficient collective communications for a new fault tolerant MPI implementation known as HARNESS [10] FT-MPI [11]. At least 2 different efforts were made in the past to improve the performance of the MPI collective communications for a given system. They either dealt with the collective communications for a specific system or tried to tune the collective communications for a given system based on mathematical models or both. Lars Paul Huse's paper on collective communications [2] studied and compared the performance of different collective algorithms on SCI based clusters. MAGPIE by Thilo Kielman et. al. [1] optimizes collective communications for clustered wide area systems. Though MAGPIE tries to find the optimum buffer size and optimum tree shape for a given collective communication on a given system, these optimum parameters are determined using a performance model called the parameterized LogP model. The MAGPIE model considered only a few network parameters for modeling collective communications. For example, it did not take into account the number of previously posted non-blocking sends, Isends, in determining the network parameters for a given message size.

In our previous work [12], [13], we built efficient algorithms for different collective communications and selected the best collective algorithm and segment

* This work was supported by the US Department of Energy through contract numberDE-FG02-99ER25378.

V.N. Alexandrov et al. (Eds.): ICCS 2001, LNCS 2073, pp. 41–50, 2001.
© Springer-Verlag Berlin Heidelberg 2001

size for a given {collective communication, number of processors, message size} tuple by experimenting with all the algorithms and all possible values for message sizes. The tuned collective communication operations were compared with various native vendor MPI implementations. The use of the tuned collective communications resulted in about 30%-650% improvement in performance over the native MPI implementations.

Although efficient, conducting the actual set of experiments to determine the optimum parameters of collective communications for a given system, was found to be time-consuming. As a first step, the best buffer size for a given algorithm for a given number of processors was determined by evaluating the performance of the algorithm for different buffer sizes. In the second phase, the best algorithm for a given message size was chosen by repeating the first phase with a known set of algorithms and choosing the algorithm that gave the best result. In the third phase, the first and second phase were repeated for different number of processors. The large number of buffer sizes and the large number of processors significantly increased the time for conducting the above experiments.

In order to reduce the time for running the actual set of experiments, the collective communications have to be modeled effectively. In this paper, we discuss the various techniques for modeling the collective communications. The reduction of time for actual experiments are achieved at 3 levels. In the first level, limited number of {collective communications, number of processors, message size} tuple combinations is explored. In the second level, the number of {algorithm, segment size} combinations for a given {collective communication, number of processors, message size} tuple is reduced. In the third level, the time needed for running an experiment for a single {collective communications, number of processors, message size, algorithm, segment size} tuple is reduced by modeling the actual experiment.

In Sect.2, we give a brief overview of our previous work regarding the automatic tuning of the collective communications. We illustrate the automatic tuning with the broadcast communication. The results in Sect.2 reiterate the usefulness of the automatic tuning approach. These results were obtained by conducting the actual experiments with all possible input parameters. In Sect.3, we describe three techniques needed for reducing the large number of actual experiments. In Sect.4, we present some conclusions. Finally in Sect.5, we outline the future direction of the research.

2 Automatically Tuned Collective Communications

A crucial step in our effort was to develop a set of competent algorithms. Table. 1 lists the various algorithms used for different collective communications.

For algorithms that involve more than one collective communication (e.g., reduce followed by broadcast in allreduce), the optimized versions of the collective communications were used. The segmentation of messages was implemented for sequential, chain, binary and binomial algorithms for all the collective communication operations.

Table 1. Collective communication algorithms

Collective Communications	*Algorithms*
Broadcast	Sequential, Chain, Binary and Binomial
Scatter	Sequential, Chain and Binary
Gather	Sequential, Chain and Binary
Reduce	Gather followed by operation, Chain, Binary, Binomial and Rabenseifner
Allreduce	Reduce followed by broadcast, Allgather followed by operation, Chain, Binary, Binomial and Rabenseifner
Allgather	Gather followed by broadcast
Allgather	Circular
Barrier	Extended ring, Distributed binomial and tournament

2.1 Results for Broadcast

The experiments consist of many phases.

Phase 1: Determining the best segment size for a given {collective operation, number of processors, message size, algorithm} tuple. The segment sizes are powers of 2, multiples of the basic data type and less than the message size.

Phase 2: Determining the best algorithm for a given {collective operation, number of processors} for each message size. Message sizes from the size of the basic data type to 1MB were evaluated.

Phase 3: Repeating phase 1 and phase 2 for different {number of processors, collective operation} combinations. The number of processors will be power of 2 and less than the available number of processors.

Our current effort is in reducing the search space involved in each of the above phases and still be able to get valid conclusions.

The experiments were conducted on four different classes of system, including clusters of Sparc and Pentium workstations and two different types of PowerPC based IBM SP2 nodes.

Fig. 1 shows the results for a tuned MPI broadcast on an IBM SP2 using "thin" nodes versus the IBM optimised vendor MPI implementation. Similar encouraging results were obtained for other systems as detailed in [12] & [13].

3 Reducing the Number of Experiments

In the experimental method described in the previous sections a large number of individual experiments have to be conducted. Even though this only needs to occur once, the time taken for all these experiments was considerable and was approximately equal to 50 hours.

The experiments conducted consist of two stages, the primary set of steps is dependent on message size, number of processors and MPI collective operation, i.e. the tuple {message size, processors, operation}. For example 64KBytes of data, 8 process broadcast. The secondary set of tests is an optimization at these

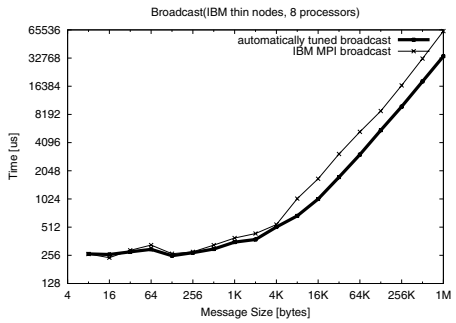


Fig. 1. Broadcast Results (IBM thin nodes)

parameters for the correct method (topology-algorithm pair) and segmentation size, i.e. the tuple {method, segment size}.

Reducing the time needed for running the actual experiments can be achieved at three different levels:

1. reducing the primary tests
2. reducing the secondary tests and
3. reducing the time for a single experiment, i.e. for a single {message size, processors, operation, method, segment size} instance.

3.1 Reducing the Primary Tests

Currently the primary tests are conducted on a fixed set of parameters, in effect making a discrete 3D grid of points. For example, varying the message size in powers of two from 8 bytes to 1 MByte, processors from 2 to 32 and the MPI operations from Broadcast to All2All etc.

This produces an extensive set of results from which accurate decisions will be made at run-time. This however makes the initial experiments time consuming and also leads to large lookup tables that have to be referenced at run time, although simple caching techniques can alleviate this particular problem.

Currently we are examining three techniques to reduce this primary set of experimental points.

1. Reduced number of grid points with interpolation. For example reducing the message size tests from {8, 16, 32, 64.. 1MB} to {8, 1024, 8192.. 1MB}.
2. Using instrumented application runs to build a table of only those collective operations that are required, i.e. not tuning operations that will never be called, or are called infrequently.
3. Using combinatorial optimizers with a reduced set of experiments, so that complex non-linear relationships between points can be correctly predicted.

3.2 Reducing the Secondary Tests

The secondary set of tests for each {message size, processors, operation} are where we have to optimize the time taken, by changing the method used (algorithm/topology) and the segmentation size (used to increase the bi-sectional bandwidth of links), i.e. {method, segment size}. Fig. 2 shows the performance of four different methods for solving an 8 processor MPI Scatter of 128KBytes of data. Several important points can be observed. Firstly, all the methods have the same basic shape that follows the form of an exponential slope followed by a plateau. Secondly, the results have multiple local optima, and that the final result (segment size equal to message size) is not usually the optimal but is close in magnitude to the optimal.

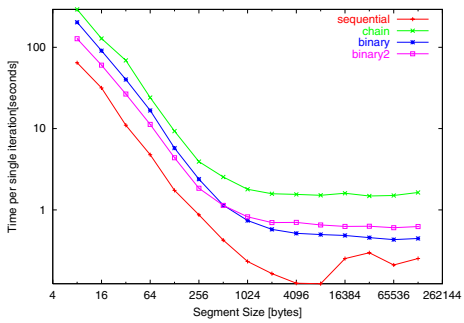


Fig. 2. Segment size verse time for various communication methods

The time taken per iteration for each method is not constant, thus many of the commonly used optimization techniques cannot be used without modification. For example in Fig. 2, a test near the largest segment size is in the order of hundreds of microseconds whereas a single test near the smallest segment size can be in the order of a 100 seconds, or two to three orders of magnitude larger.

For this reason we have developed two methods that reduce the search space to tests close to the optimal values, and a third that runs a full set of segment-size tests on only a partial set of nodes.

The first two methods use a number of different hill descent algorithms known as the Modified Gradient Descent MGD and the Scanning Modified Gradient Descent (SMGD) that are explained in [13]. They primarily reduce the search times by searching the least expensive (in time) search spaces first while performing various look ahead algorithms to avoid non optimal minima. Using these two methods the time to find the optimal segment size for the scatter show in Fig. 2 is reduced from 12613 seconds to just 39 seconds or a speed up of 318.

The third method used to reduce tests is based on the relationship between some performance metrics of a collective that utilizes a tree topology and those of a pipeline that is based only on the longest edge of the tree as shown in Fig. 3. In particular the authors found that the pipeline can be used to find the

optimal segmentation size at greatly reduced time as only a few nodes need to be tested as opposed to the whole tree structure. For the 128 KB 8 process scatter discussed above, an optimal segment size was found in around 1.6 seconds per class of communication method (such as tree, sequential or ring). i.e. 6.4 seconds versus 39 for the gradient descent methods on the complete topologies or 12613 for the complete exhaustive search.

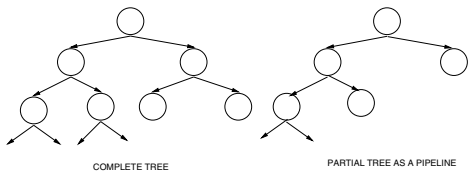


Fig. 3. The Pipeline Model

3.3 Reducing the Single-Experiment Time

Running the actual experiments to determine the optimized parameters for collective communications is time-consuming due to the overheads associated with the startup of different processes, setting up of the actual data buffers, communication of messages between different processes etc.. We are building experimental models that simulate the collective algorithms but incur less time to execute than the actual experiments. As part of this approach, we discuss the modeling experiments for broadcast in the following sub sections.

General Overview. All the broadcast algorithms are based on a common methodology. The root in the broadcast tree continuously does non-blocking sends of MPI, MPI_Isends, to send individual message buffers to its children. The other nodes post all their non-blocking receives of MPI, MPI_Irecvs, initially. The nodes between the root node and the leaf nodes in the broadcast tree, send a segment to their children as soon as the segment is received.

After determining the times for individual Isends and the times for message receptions, a broadcast schedule as illustrated by Fig. 4 can be used to predict the total completion time for the broadcast.

A broadcast schedule such as the one shown in Fig. 4 can be used to accurately model the overlap in communications, a feature that was lacking in the parameterized LogP model [1].

Measurement of PointPoint Communications. As observed in the previous section, accurate measurements of the time for Isends and the time for the reception of the messages are necessary for efficient modeling of broadcast operations. Previous communications models [3], [1], do not efficiently take into

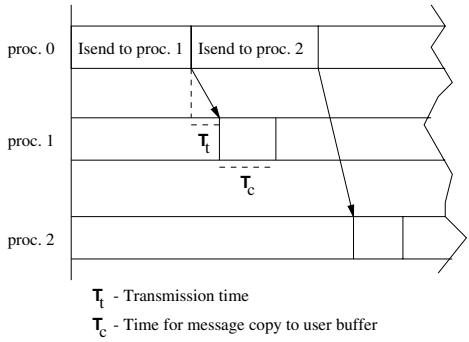


Fig. 4. Illustration of Broadcast Schedule

account the different types of Isends. Also, these models overlook the fact that the performance of an Isend can vary depending on the number of Isends posted previously. Thus the parameters, the send overhead, *os(m)*, the receive overhead, *or(m)*, the gap value, *g(m)*, for a given message size m, that were discussed in the parameterized LogP model can vary from a particular point in execution to another depending on the number of pending Isends and the type of the Isend.

MPI implementations employ different types of Isends depending on the size of the message transmitted. The popular modes of Isends are blocking, immediate and randezevous and are illustrated by Fig. 5

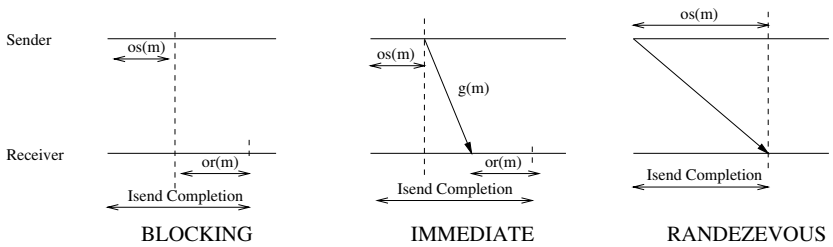


Fig. 5. Di erent modes for Isends

The parameters associated with the different modes of Isends can vary depending the number of Isends posted earlier. Hence, for example, in the case of immediate mode, the Isends can lead to overflow of buffer space in the receive end, which will eventually result in larger g(m) and os(m).

A simple model. In this section, we describe a simple model that we have built to calculate the performance of collective communications. At this point, the model is not expected to give good predictions of the performance. A study of the results of this primitive model is useful in understanding the complexities

of Isends and developing some insights on building a better model for collective communications.

The model uses the data for sender overhead, *os(m)*, receiver overhead, *or(m)* and gap value, *g(m)* for the different types of Isends show in Fig. 5.But the model does not use the value of g(m) effectively and it assumes that multiple messages to a node can be sent continuously. The model also does not take into account the number of Isends previously posted.

The send overhead, *os(m)* is determined for different message sizes by observing the time taken for the corresponding Isends. The time for Isends, *os(m)*, increases as the message size is increased upto a certain message size beyond which, os(m), falls to a small value. At this message size, the Isend switches from the blocking to immediate mode. *or(m)* for blocking mode is determined by allowing the receiver to post a blocking receive after making sure the message has been transmitted over the network to the receiver end and determining the time taken for the blocking receive. In the immediate mode, the sender has to wait for *g(m)* before transmitting the next message. This time is determined by posting an Isend and determining the time taken for the subsequent Wait. In the immediate mode, *or(m)+g(m)*, is calculated by determining the time for a ping-pong transmission between a sender and a receiver and subtracting 2**os(m)* from the ping-pong time. For each of the above experiments, 10 different runs were made and averages were calculated. The experiments were repeated at different points in time on shared machines and the standard deviation was found to be as low as 40.

With these simplifying assumptions, the model builds a broadcast schedule for flat, chain, binary and binomial broadcast trees for 2, 4, 8 and 16 processors. Fig. 6 compares the actual and predicted broadcast times for a flat tree broadcast sending a 128K byte message using 8 processors on a Solaris workstation.

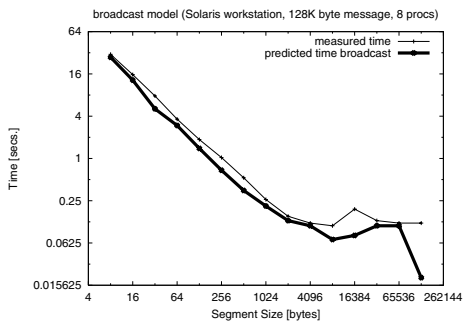


Fig. 6. Flat Tree broadcast

While the model gives good predictions for smaller segment sizes or larger number of segments, it underestimates for smaller number of segments. Also, the performance is poor if the message between the nodes is transmitted as only one

segment. For a segment size of 128K, the Isend switches to immediate mode. Since the system has to buffer the messages for immediate Isends, the buffer capacity acts as a bottleneck as the number of posted Isends increase. Since the model does not take into account the number of Isends posted, it gives poor performance for 128K byte messages.

Fig. 7 compares the actual and predicted broadcast times for a chain tree broadcast sending a 128K byte message using 8 processors on the same system.

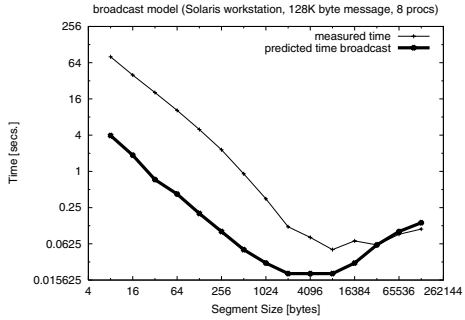


Fig. 7. Chain tree broadcast

Since the model assumes that messages to a single node can be sent continuously, and since in a chain broadcast tree, the segments are sent continuously to a single node, the model gives much smaller times than the actual times for smaller segment size or for large number of segments.

From the above experiments, we recognize that good models for predicting collective communications have to take into account all the possible scenarios for sends and receives in order to build a good broadcast schedule. While our simplified model did not give good predictions for the results shown, it helped to identify some of the important factors that have to be taken into account for efficient modeling.

4 Conclusion

Modeling the collective communications to determine the optimum parameters of the collective communications is a challenging task, involving complex scenarios. A single simplified model will not be able to take into account the complexities associated with the communications. A multi-dimensional approach towards modeling, where various tools for modeling are provided to the user to accurately model the collective communications on his system, is necessary. Our techniques regarding the reduction of number of experiments are steps towards constructing the tools for modeling. These techniques have given promising results and have helped identify the inherent complexities associated with the collective communications.

5 Future Work

While our initial results are promising and provide us some valuable insights regarding collective communications, much work still has to be done to provide comprehensive set of techniques for modeling collective communications. Selecting the right set of techniques for modeling based on the system dynamics is an interesting task and will be explored further.

References

1. Thilo Kielmann, Henri E. Bal and Segei Gorlatch. Bandwidth-efficient Collective Communication for Clustered Wide Area Systems.*IPDPS 2000*, Cancun , Mexico. (May 1-5, 2000)
2. Lars Paul Huse. Collective Communication on Dedicated Clusters of Workstations.*Proceedings of the 6th European PVM/MPI Users' Group Meeting*, Barcelona, Spain, Spetmeber 1999. p(469-476).
3. David Culler, R. Karp, D. Patterson, A. Sahay, K.E. Schauser, E. Santos , R. Subramonian and T. von Eicken. LogP: Towards a Realistic Model of Parallel Computation. In *Proc. Symposium on Principles and Practice of Parallel Programming* (PpoPP), pages 1-12, San Diego, CA (May 1993).
4. R. Rabenseifner. A new optimized MPI reduce algorithm. http://www.hlrs.de/structure/support/parallel_computing/models/mpi/myreduce.html (1997).
5. Marc Snir, Steve Otto, Steven Huss-Lederman, David Walker and Jack Dongarra. MPI- The Complete Reference. *Volume 1, The MPI Core, second edition* (1998).
6. M. Frigo. FFTW: An Adaptive Software Architecture for the FFT. *Proceedings of the ICASSP Conference*, page 1381, Vol. 3. (1998).
7. R. Clint Whaley and Jack Dongarra. Automatically Tuned Linear Algebra Software. *SC98: High Performance Networking and Computing.* http://www.cs.utk.edu/ rwhaley/ATL/INDEX.HTM. (1998)
8. L. Prylli and B. Tourancheau. "BIP: a new protocol designed for high performance networking on myrinet". In the *PC-NOW workshop, IPPS/SPDP 1998*, Orlando, USA, 1998.
9. Debra Hensgen, Raphael Finkel and Udi Manber. Two algorithms for Barrier Synchroniztion. *International Journal of Parallel Programming*, Vol. 17, No. 1, 1988.
10. M. Beck, J. Dongarra, G. Fagg, A. Geist, P. Gray, J.Kohl, M. Migliardi, K. Moore, T. Moore, P. Papadopoulous, S. Scott, V. Sunderam,"HARNESS: a next generation distributed virtual machine"", *Journal of Future Generation Computer Systems*, (15), Elsevier Science B.V., 1999.
11. Graham E. Fagg and Jack J. Dongarra, "FT-MPI: Fault Tolerant MPI, Supporting Dynamic Applications in a Dynamic World", *Proc. of EuroPVM-MPI 2000, Lecture notes in Computer Science*, Vol. 1908, pp346-353, Springer Verlag, 2000.
12. Graham E. Fagg, Sathish S. Vadhiyar, Jack J. Dongarra, "ACCT: Automatic Collective Communications Tuning", *Proc of EuroPVM-MPI 2000, Lecture Notes in Computer Science*, Vol. 1908, pp354-361, Springer Verlag, 2000.
13. Sathish S. Vadhiyar, Graham E. Fagg, Jack J. Dongarra, "Automatically Tuned Collective Communications", *Proceedings of SuperComputing2000*, Dallas, Texas, Nov. 2000.

A Family of High-Performance Matrix Multiplication Algorithms

John A. Gunnels[1], Greg M. Henry[2], and Robert A. van de Geijn[1]

[1] Department of Computer Sciences, The University of Texas, Austin, TX 78712,
{gunnels,rvdg}@cs.utexas.edu,
WWW home page: http://www.cs.utexas.edu/users/{gunnels,rvdg}/
[2] Intel Corp., Bldg EY2-05, 5350 NE Elam Young Pkwy, Hillsboro, OR 97124-6461,
greg.henry@intel.com,
WWW home page: **http://www.cs.utk.edu/∼ghenry/**

Abstract. During the last half-decade, a number of research efforts have centered around developing software for generating automatically tuned matrix multiplication kernels. These include the PHiPAC project and the ATLAS project. The software end-products of both projects employ brute force to search a parameter space for blockings that accommodate multiple levels of memory hierarchy. We take a different approach: using a simple model of hierarchical memories we employ mathematics to determine a locally-optimal strategy for blocking matrices. The theoretical results show that, depending on the shape of the matrices involved, different strategies are locally-optimal. Rather than determining a blocking strategy at library generation time, the theoretical results show that, ideally, one should pursue a heuristic that allows the blocking strategy to be determined dynamically at run-time as a function of the shapes of the operands. When the resulting family of algorithms is combined with a highly optimized inner-kernel for a small matrix multiplication, the approach yields performance that is superior to that of methods that automatically tune such kernels. Preliminary results, for the Intel Pentium (R) III processor, support the theoretical insights.

1 Introduction

Research in the development of linear algebra libraries has recently shifted to the automatic generation and optimization of the matrix multiplication kernels. The underlying idea is that many linear algebra operations can be implemented in terms of matrix multiplication [2,10,6] and thus it is this operation that should be highly optimized on different platforms. Since the coding effort required to achieve this is considerable, especially when multiple layers of cache are involved, the general consensus is that this process should be automated.

In this paper, we develop a theoretical framework that (1) suggests a formula for the block sizes that should be used at each level of the memory hierarchy, and (2) restricts the possible loop orderings to a specific family of algorithms for matrix multiplication. We show how to use these results to build highly optimized matrix multiplication implementations that utilize the caches in a locally-optimal fashion. The results could be equally well used to limit the search space that must be examined by packages that automatically tune such kernels.

V.N. Alexandrov et al. (Eds.): ICCS 2001, LNCS 2073, pp. 51–60, 2001.
© Springer-Verlag Berlin Heidelberg 2001

The current pursuit of highly optimized matrix kernels constructed by coding in a high-level programming language started with the implementation of the FORTRAN implementation of Basic Linear Algebra Subprograms (BLAS) [4] for the IBM POWER2 (TM) [1]. Subsequently, the PHiPAC project [3] demonstrated that high-performance matrix multiplication kernels can be written in C and that code generators could be used to automatically generate many different blockings, allowing automatic tuning. Next, the ATLAS project [11] extended these ideas by reducing the kernel that is called once matrices are massaged to be in the L1 cache into one specific case: $C = A^T B + \beta C$ for small matrices A, B, and C and by reducing the space searched for optimal blockings. Furthermore it marketed the methodology allowing it to gain wide-spread acceptance and igniting the current trend in the linear algebra community towards automatically tuned libraries. Finally, there has been a considerable recent interest in recursive algorithms and recursive data structures. The idea here is that by recursively partitioning the operands, blocks that fit in the different levels of the caches will automatically be encountered [8]. By storing matrices recursively, blocks that are encountered during the execution of the recursive algorithms will be in contiguous memory [7,9].

Other work closely related to this topic is discussed in other papers presented as part of this session of the conference.

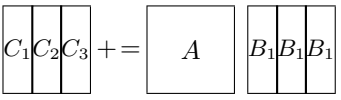
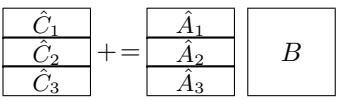
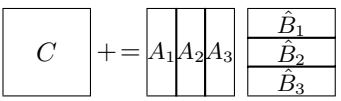
2 Notation and Terminology

2.1 Special Cases of Matrix Multiplication

The general form of a matrix multiply is $C \leftarrow \alpha AB + \beta C$ where C is $m \times n$, A is $m \times k$, and B is $k \times n$. We will use the following terminology when referring to a matrix multiply when two dimensions are large and one is small:

	Condition	Shape	
Matrix-panel multiply	n is small	$C = A\,B + C$	(1)
Panel-matrix multiply	m is small	$C = A\,B + C$	(2)
Panel-panel multiply	k is small	$C = A\,B + C$	(3)

The following observation will become key to understanding concepts encountered in the rest of the paper: Partition $X = \left(X_1 \middle| \cdots \middle| X_{N_X} \right) = \left(\begin{array}{c} \hat{X}_1 \\ \hline \vdots \\ \hline \hat{X}_{M_X} \end{array} \right)$ for $X \in \{A, B, C\}$, where C_j is $m \times n_j$, $\hat{C}_i$ is $m_i \times n$, A_p is $m \times k_p$, $\hat{A}_i$ is $m_i \times k$, B_j is $k \times n_j$, and $\hat{B}_p$ is $k_p \times n$. Then $C \leftarrow AB + C$ can be achieved by

multiple matrix-panel multiplies:	$C_j \leftarrow AB_j + C_j$ for $j = 1, \ldots, N_C$	
multiple panel-matrix multiplies:	$\hat{C}_i \leftarrow \hat{A}_i B + \hat{C}_i$ for $i = 1, \ldots, M_C$	
multiple panel-panel multiplies	$C \leftarrow \sum_p^{N_A} A_p \hat{B}_p + C$	

2.2 A Cost Model for Hierarchical Memories

The memory hierarchy of a modern microprocessor is often viewed as a pyramid: At the top of the pyramid, there are the processor registers, with extremely fast access. At the bottom, there are disks and even slower media. As one goes down the pyramid, while the financial cost of memory decreases, the amount of memory increases along with the time required to access that that memory.

We will model the above-mentioned hierarchy naively as follows: (1) The memory hierarchy consists of H levels, indexed $0, \ldots, H-1$. Level 0 corresponds to the registers. We will often denote the ith level by L_i. Notice that on a typical current architecture L_1 and L_2 correspond the level 1 and level 2 data caches and L_3 corresponds to RAM. (2) Level h of the memory hierarchy can store S_h floating-point numbers. Generally $S_0 \leq S_1 \leq \cdots \leq S_{H-1}$. (3) Loading a floating-point number stored in level $h+1$ to level h costs time ρ_h. We will assume that $\rho_0 < \rho_1 < \cdots < \rho_{H-1}$. (4) Storing a floating-point number from level h to level $h+1$ costs time σ_h. We will assume that $\sigma_0 < \sigma_1 < \cdots < \sigma_{H-1}$. (5) If $m_h \times n_h$ matrix C, $m_h \times k_h$ matrix A, and $k_h \times n_h$ matrix B are all stored in level h of the memory hierarchy then forming $C \leftarrow AB + C$ costs time $2m_h n_h k_h \gamma_h$. (Notice that γ_h will depend on m_h, n_h, and k_h).

3 Building-Blocks for Matrix Multiplication

Consider the matrix multiplication $C \leftarrow AB + C$ where $m_{h+1} \times n_{h+1}$ matrix C, $m_{h+1} \times k_{h+1}$ matrix A, and $k_{h+1} \times n_{h+1}$ matrix B are all stored in L_{h+1}. Let us assume that somehow an efficient matrix multiplication kernel exists for matrices stored in L_h. In this section, we develop three distinct approaches for matrix multiplication kernels for matrices stored in L_{h+1}.

Partition

$$C = \left(\begin{array}{c|c|c} C_{11} & \cdots & C_{1N} \\ \hline \vdots & & \vdots \\ \hline C_{M1} & \cdots & C_{MN} \end{array}\right), A = \left(\begin{array}{c|c|c} A_{11} & \cdots & A_{1K} \\ \hline \vdots & & \vdots \\ \hline A_{M1} & \cdots & A_{MK} \end{array}\right), \text{ and } B = \left(\begin{array}{c|c|c} B_{11} & \cdots & B_{1N} \\ \hline \vdots & & \vdots \\ \hline B_{K1} & \cdots & B_{KN} \end{array}\right) \tag{4}$$

Algorithm 1

for $j = 1, \ldots, N$	
for $i = 1, \ldots, M$	
Load C_{ij} from L_{h+1} to L_h.	$m_h n_h \rho_h$
for $p = 1, \ldots, K$	
Load A_{ip} from L_{h+1} to L_h.	$m_h k_h \rho_h$
Load B_{pj} from L_{h+1} to L_h.	$k_h n_h \rho_h$
Update $C_{ij} \leftarrow A_{ip} B_{pj} + C_{ij}$	$2 m_h n_h k_h \gamma_h$
endfor	
Store C_{ij} from L_h to L_{h+1}	$m_h n_h \sigma_h$
endfor	
endfor	

Fig. 1. Multiple panel-panel multiply based blocked matrix multiplication.

where C_{ij} is $m_h \times n_h$, A_{ip} is $m_h \times k_h$, and B_{pj} is $k_h \times n_h$. The objective of the game will be to determine optimal m_h, n_h, and k_h.

3.1 Multiple Panel-Panel Multiplies in L_h

Noting that $C_{ij} \leftarrow \sum_{p=1}^{K} A_{ip} B_{pj} + C_{ij}$, let us consider the algorithm in Fig. 1 for computing the matrix multiplication. In that figure the costs of the various operations are shown to the right. The order of the outer-most loops is irrelevant to the analysis.

The cost for updating C is given by

$$m_{h+1} n_{h+1} (\rho_h + \sigma_h) + m_{h+1} n_{h+1} k_{h+1} \frac{\rho_h}{n_h} + m_{h+1} n_{h+1} k_{h+1} \frac{\rho_h}{m_h} + 2 m_{h+1} n_{h+1} k_{h+1} \gamma_h$$

Since it equals $2 m_{h+1} n_{h+1} k_{h+1}$, solving for γ_{h+1}, the effective cost per floating-point operation at level L_{h+1}, yields $\gamma_{h+1}^{PP} = \frac{\rho_h + \sigma_h}{2 k_{h+1}} + \frac{\rho_h}{2 n_h} + \frac{\rho_h}{2 m_h} + \gamma_h$. The question now is how to find the m_h, n_h, and k_h that minimize γ_{h+1} under the constraint that C_{ij}, A_{ik} and B_{kj} all fit in L_h, i.e., $m_h n_h + m_h k_h + n_h k_h \leq S_h$. The smaller k_h, the more space in L_h can be dedicated to C_{ij} and thus the smaller the fractions ρ_h / m_h and ρ_h / n_h can be made. A good strategy is thus to let essentially all of L_h be dedicated to C_{ij}, i.e., $m_h n_h \approx S_h$. The minimum is then attained when $m_h \approx n_h \approx \sqrt{S_h}$.

Notice that it suffices to have $m_{h+1} = m_h$ or $n_{h+1} = n_h$ for the above cost of γ_{h+1} to be minimized. Thus, the above already holds for the special cases depicted in (1) and (2), i.e., when $N = 1$ and $M = 1$ in (4), respectively.

The innermost loop in Alg. 1 implements multiple panel-panel multiplies since k_h is assumed to be small relative to m_h and n_h. Hence the name of this section.

3.2 Multiple Matrix-Panel Multiplies in L_h

Moving the loops over l and i to the outside we obtain the algorithm in Fig. 2 (left). Performing an analysis similar to that given in Section 3.1 the

Algorithm 2

for $p = 1, \ldots, K$
  for $i = 1, \ldots, M$
    Load A_{ip} from L_{h+1} to L_h.
    for $j = 1, \ldots, N$
      Load C_{ij} from L_{h+1} to L_h.
      Load B_{pj} from L_{h+1} to L_h.
      Update $C_{ij} \leftarrow A_{ip}B_{pj} + C_{ij}$
      Store C_{ij} from L_h to L_{h+1}
    endfor
  endfor
endfor

Algorithm 3

for $j = 1, \ldots, N$
  for $p = 1, \ldots, K$
    Load B_{pj} from L_{h+1} to L_h.
    for $i = 1, \ldots, M$
      Load C_{ij} from L_{h+1} to L_h.
      Load A_{ip} from L_{h+1} to L_h.
      Update $C_{ij} \leftarrow A_{ip}B_{pj} + C_{ij}$
      Store C_{ij} from L_h to L_{h+1}
    endfor
  endfor
endfor

Fig. 2. Multiple matrix-panel (left) and panel-matrix (right) multiply based blocked matrix multiplication.

effective cost of a floating-point operation is now given by $\gamma_{h+1}^{MP} = \frac{\rho_h}{2n_{h+1}} + \frac{\rho_h+\sigma_h}{2k_h} + \frac{\rho_h}{2m_h} + \gamma_h$.

Again, the question is how to find the m_h, n_h, and k_h that minimize γ_{h+1} under the constraint that C_{ij}, A_{ik} and B_{kj} all fit in L_h, i.e., $m_h n_h + m_h k_h + n_h k_h \leq S_h$. Note that the smaller n_h, the more space in L_h can be dedicated to A_{il} and thus the smaller the fractions $(\rho_h + \sigma_h)/2k_h$ and $\rho_h/2m_h$ can be made. A good strategy is thus to let essentially all of L_h be dedicated to A_{il}, i.e., $m_h k_h \approx S_h$. The minimum is then attained when $m_h \approx k_h \approx \sqrt{S_h}$.

Notice that it suffices to have $m_{h+1} = m_h$ or $k_{h+1} = k_h$ for the above cost of γ_{h+1} to be minimized. In other words, the above holds for the special cases depicted in (2) and (3), i.e., when $M = 1$ and $K = 1$ in (4), respectively.

The innermost loop in Alg. 2 implements multiple matrix-panel multiplies since n_h is small relative to m_h and k_h. Thus the name of this section.

3.3 Multiple Panel-Matrix Multiplies in L_h

Finally, moving the loops over p and j to the outside we obtain the algorithm given in Fig. 2 (right). This time, the effective cost of a floating-point operation is given by $\gamma_{h+1}^{PM} = \frac{\rho_h}{2m_{h+1}} + \frac{\rho_h+\sigma_h}{2k_h} + \frac{\rho_h}{2n_h} + \gamma_h$.

Again, the question is how to find the m_h, n_h, and k_h that minimize γ_{h+1} under the constraint that C_{ij}, A_{ik} and B_{kj} all fit in L_h, i.e., $m_h n_h + m_h k_h + n_h k_h \leq S_h$. Note that the smaller m_h, the more space in L_h can be dedicated to B_{pj} and thus the smaller the fractions $(\rho_h + \sigma_h)/2k_h$ and $\rho_h/2n_h$ can be made. A good strategy in this case is to dedicate essentially all of L_h to B_{pj}, i.e., $n_h k_h \approx S_h$. The minimum is then attained when $n_h \approx k_h \approx \sqrt{S_h}$.

Notice that it suffices to have $n_{h+1} = n_h$ and/or $k_{h+1} = k_h$ for the above cost of γ_{h+1} to be achieved. In other words, the above holds for the special cases depicted in (1) and (3), i.e., when $N = 1$ and $K = 1$ in (4), respectively.

3.4 Summary

The conclusions to draw from Sections 2.1 and 3.1–3.3 are: (1) There are three shapes of matrix multiplication that one expects to encounter at each level of the memory hierarchy: panel-panel, matrix-panel, and panel-matrix multiplication. (2) If one such shape is encountered at L_{h+1}, a locally-optimal approach to utilizing L_h will perform multiple instances with one of the other two shapes. (3) Given that multiple instances of a given shape are to be performed, the strategy is to move a submatrix of one of the three operands into L_h (we will call this the *resident* matrix in L_h), filling most of that layer, and to amortize the cost of this data movement by streaming submatrices from the other operands from L_{h+1} to L_h.

Interestingly enough, the shapes discussed are exactly those that we encountered when studying a class of matrix multiplication algorithms on distributed memory architectures [5]. This is not surprising, since distributed memory is just another layer in the memory hierarchy.

4 A Family of Algorithms

We now turn the observations made above into a practical implementation.

High-performance implementations of matrix multiplication typically start with an "inner-kernel". This kernel carefully orchestrates the movement of data in and out of the registers and the computation under the assumption that one or more of the operands are in the L1 cache. For our implementation on the Intel Pentium (R) III processor, the inner-kernel performs the operation $C = A^T B + \beta C$ where 64×8 matrix A is kept in the L1 cache. Matrices B and C have a large number of columns, which we view as multiple-panels, with each panel of width one. Thus, our inner-kernel performs a multiple matrix-panel multiply (MMP) with a transposed resident matrix A. The technical reasons why this particular shape was selected go beyond the scope of this paper.

While it may appear that we thus only have one of the three kernels for operation in the L1 cache, notice that for the submatrices with which we compute at that level one can instead compute $C^T = B^T A + C^T$, reversing the role of A and B. This simple observation allows us to claim that we also have an inner-kernel that performs a multiple panel-matrix multiply (MPM).

Let us introduce a naming convention for a family of algorithms that perform the discussed algorithms at different levels of the memory hierarchy:

$$<\text{kernel at } L_3>\text{-}<\text{kernel at } L_2>\text{-}<\text{kernel at } L_1>.$$

For example MPP-MPM-MMP will indicate that the L_3-kernel uses multiple panel-panel multiplies, calls the L_2-kernel that uses multiple matrix-panel multiplies, which in turn calls the L_1-kernel that uses multiple panel-matrix multiplies. Given the constraint that only two of the possible three kernel algorithms are implemented at L_1, the tree of algorithms in Fig. 3 can be constructed.

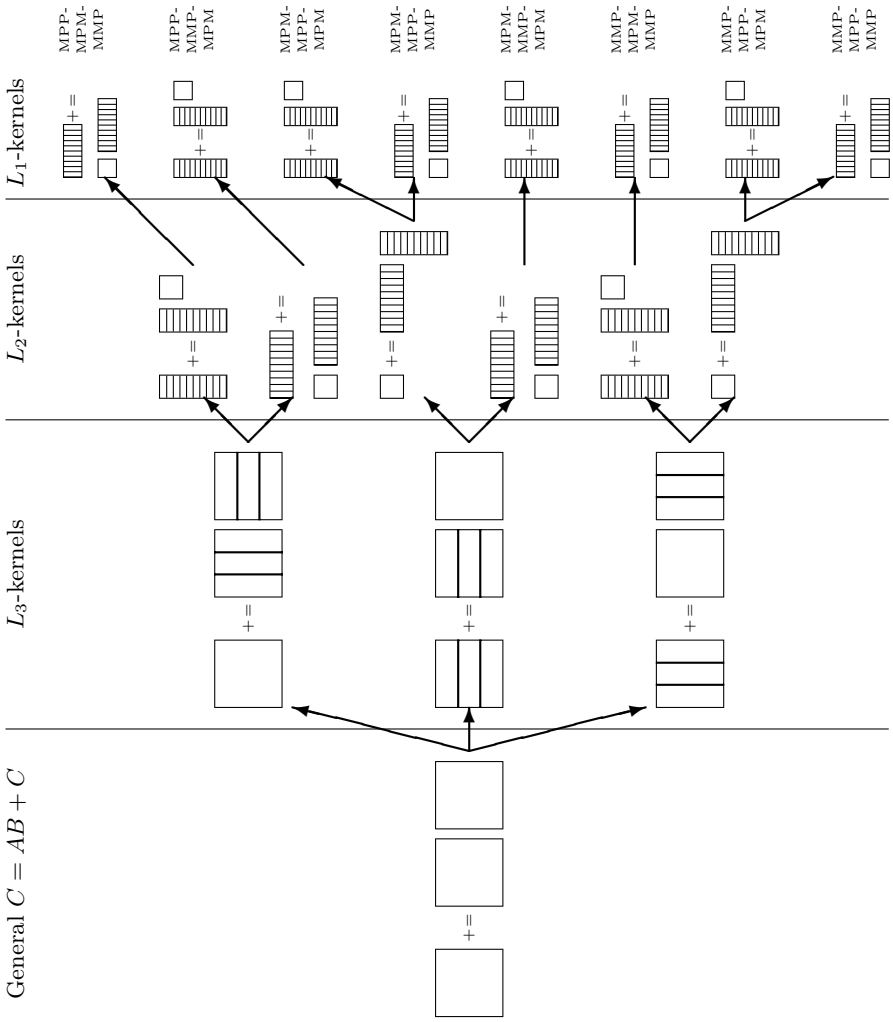
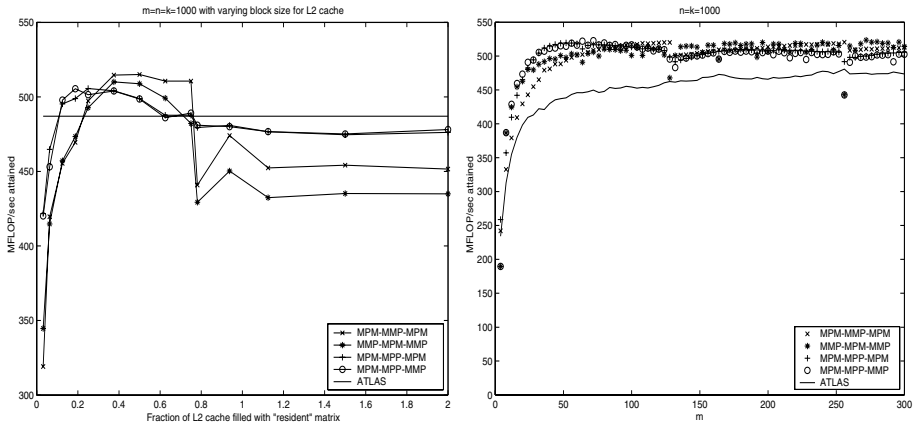


Fig. 3. Possible algorithms for matrices in memory level L_3 given all L_2-kernels.



(a) (b)

Fig. 4. Left: Performance for xed dimensions $m = n = k = 1000$ as a function of the size of the resident matrix in the L2 cache. Right: Performance as a function of m when $n = k = 1000$.

5 Performance

In this section, we report performance attained by the different algorithms. Performance is reported by the rate of computations attained, in millions of floating-point operations per second (MFLOP/sec) using 64-bit arithmetic. For the usual matrix dimensions m, n, and k, we use the operation count $2mnk$ for the matrix multiplication. We tested performance of the operation $C = C - AB$ ($\alpha = -1$ and $\beta = 1$) since this is the case most frequently encountered when matrix multiplication is used in libraries such as LAPACK.

We report performance on an Intel Pentium (R) III (650 MHz) processor with a 16 Kbyte L1 data cache and a 256 Kbyte L2 cache running RedHat Linux 6.2. The inner-kernel, which perform the operation $C \leftarrow A^T B + \beta C$ with 64×8 matrix A and $64 \times k$ matrix B, was hand-coded using Intel Streaming SIMD Extensions (TM) (SSE). In order to keep the graphs readable, we only report performance for four of the eight possible algorithms. For reference, we report performance of the matrix multiply from ATLAS R3.2 ,which does not use Intel SSE instructions, for this architecture.

Our first experiment is intended to demonstrate that the block size selected for the matrix that remains resident in the L2 cache has a clear effect on the overall performance of the matrix multiplication routine. In Fig. 4(a) we report performance attained as a function of the fraction of the L2 cache filled with the resident matrix when a matrix multiplication with $k = m = n = 1000$ is executed. This experiment tests our theory that reuse of data in the L2 cache impacts overall performance as well as our theory that the resident matrix should occupy "most" of the L2 cache. Note that performance improves as a larger fraction of the L2 cache is filled with the resident matrix. Once the resident matrix fills more than half of the L2 cache, performance starts to deminish.

This is consistent with the theory which tells us that some of the cache must be used for the matrices that are being streamed from main memory. Once more than 3/4 of the L2 cache is filled with the resident matrix, performance drops significantly. This is consistent with the scenario wherein parts of the other matrices start evicting parts of the resident matrix from the L2 cache. Based on the above experiment, we fix the block size for the resident matrix in the L2 cache to 128×128, which fills exactly half of this cache, for the remaining experiments.

In Fig. 4(b) we show performance as a function of m when n and k are fixed to be large. There is more information in this graph than we can discuss in this paper. Notice for example that performance of the algorithm that performs multiple panel-matrix multiplies in the L_3 cache and multiple matrix-panel multiplies in the L_2 cache, `MPM_MMP_MPM`, increases as m increases to a multiple of 128. This is consistent with the theory.

For additional and more up-to-date performance graphs, and related discussion, we refer the reader to the ITXGEMM web page mentioned in the conclusion.

6 Conclusion

In this paper, theoretical insight was used to motivate a family of algorithms for matrix multiplication on hierarchical memory architectures. The approach attempts to amortize the cost of moving data between memory layers in a fashion that is locally-optimal. Preliminary experimental results on the Intel Pentium (R) III processor appear to support the theoretical results.

Many questions regarding this subject are not addressed in this paper, some due to space limitations. For example, the techniques can be, and have been, trivially extended to the other cases of matrix multiplication: $C \leftarrow \alpha A^T B + \beta C$, $C \leftarrow \alpha A B^T + \beta C$, and $C \leftarrow \alpha A^T B^T + \beta C$ by transposing matrices at appropriate stages in the algorithm. Also, while we claim that given different matrix dimensions, m, n, and k, a different algorithm may be best, we do not address how to choose from the different algorithms. We have developed simple heuristics that yield very satisfactory results. Experiments that support the theory, performed on a number of different architectures, are needed to draw definitive conclusions. The theory should be extended to include a model of cache-replacement policies. How performance is affected by the hand-coded inner-kernel needs to be quantified. We hope to address these issues in a future paper.

Clearly, our techniques can be used to reduce the set of block sizes to be searched at each level of the memory hierarchy. Thus, our techniques could be combined with techniques for automatically generating the inner-kernel and/or an automated search for the optimal block sizes.

More information: `http://www.cs.utexas.edu/users/flame/ITXGEMM/`.

Acknowledgments: We thank Dr. Fred Gustavson for valuable feedback regarding this project.

References

1. R.C. Agarwal, F.G. Gustavson, and M. Zubair. Exploiting functional parallelism of POWER2 to design high-performance numerical algorithms. *IBM Journal of Research and Development*, 38(5), Sept. 1994.
2. E. Anderson, Z. Bai, C. Bischof, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, S. Ostrouchov, and D. Sorensen. *LAPACK Users' Guide - Release 2.0.* SIAM, 1994.
3. J. Bilmes, K. Asanovic, C.W. Chin, and J. Demmel. Optimizing matrix multiply using PHiPAC: a portable, high-performance, ANSI C coding methodology. In *Proceedings of the International Conference on Supercomputing.* ACM SIGARC, July 1997.
4. Jack J. Dongarra, Jeremy Du Croz, Sven Hammarling, and Iain Duff. A set of level 3 basic linear algebra subprograms. *ACM Trans. Math. Soft.*, 16(1):1–17, March 1990.
5. John Gunnels, Calvin Lin, Greg Morrow, and Robert van de Geijn. A flexible class of parallel matrix multiplication algorithms. In *Proceedings of First Merged International Parallel Processing Symposium and Symposium on Parallel and Distributed Processing (1998 IPPS/SPDP '98)*, pages 110–116, 1998.
6. John A. Gunnels and Robert A. van de Geijn. Formal methods for high-performance linear algebra libraries. In Ronald F. Boisvert and Ping Tak Peter Tang, editors, *The Architecture of Scientific Software.* Kluwer Academic Press, 2001.
7. F. Gustavson, A. Henriksson, I. Jonsson, B. Kågström, and P. Ling. Recursive blocked data formats and BLAS's for dense linear algebra algorithms. In B. Kågström et al., editor, *Applied Parallel Computing, Large Scale Scientific and Industrial Problems*, volume 1541 of *Lecture Notes in Computer Science*, pages 195–206. Springer-Verlag, 1998.
8. F. G. Gustavson. Recursion leads to automatic variable blocking for dense linear-algebra algorithms. *IBM Journal of Research and Development*, 41(6):737–755, November 1997.
9. Greg Henry. BLAS based on block data structures. Theory Center Technical Report CTC92TR89, Cornell University, Feb. 1992.
10. B. Kågström, P. Ling, and C. Van Loan. GEMM-based level 3 BLAS: High performance model implementations and performance evaluation benchmark. Technical Report CS-95-315, Univ. of Tennessee, Nov. 1995.
11. R. Clint Whaley and Jack J. Dongarra. Automatically tuned linear algebra software. In *Proceedings of SC98*, Nov. 1998.

Performance Evaluation of Heuristics for Scheduling Pipelined Multiprocessor Tasks

M. Fikret Ercan[1], Ceyda Oguz[2], and Yu-Fai Fung[3]

[1] School of Electrical and Electronic Engineering, Singapore Polytechnic, Singapore
mfercan@sp.edu.sg

[2] Department of Management, The Hong Kong Polytechnic University, Hong Kong S.A.R.
msceyda@polyu.edu.hk

[3] Department of Electrical Eng., The Hong Kong Polytechnic University, Hong Kong S.A.R.
eeyffung@polyu.edu.hk

Abstract. This paper presents the evaluation of the solution quality of heuristic algorithms developed for scheduling multiprocessor tasks in a class of multiprocessor architecture designed for real-time operations. MIMD parallelism and multiprogramming support are the two main characteristics of multiprocessor architecture considered. The solution methodology includes different techniques including simulated annealing, tabu search, as well as well-known simple priority rule based heuristics. The results obtained by these different techniques are analyzed for different number of jobs and machine configurations.

1 Introduction

In order to cope with the computing requirements in many real-time applications, such as machine vision, robotics, and power system simulation, parallelism in two directions, space (data or control) and time (temporal), are exploited simultaneously [4, 10]. Multitasking computing platforms are particularly developed to exploit this computing structure. These architectures provide either a pool of processors that can be partitioned into processor clusters or processor arrays prearranged in multiple layers. PASM [11], NETRA[2] or IUA [12] are the examples to such architectures. In both approaches a communication mechanism is provided among processor clusters to support pipelining of tasks. The computing platform achieves multi-tasking (or multiprogramming) by allowing simultaneous execution of independent parallel algorithms in independent processor groups. This class of computers is specially developed for applications where operations are repetitive. A good example to this computing structure is real-time computer vision, where overall structure is made of a stream of related tasks. Operations performed on each image frame can be categorized as low, intermediate and high level. The result of an algorithm in low level initiates another algorithm in intermediate level and so on. By exploiting available spatial parallelism, algorithms at each level can be split into smaller grains to reduce their computation

V.N. Alexandrov et al. (Eds.): ICCS 2001, LNCS 2073, pp. 61-70, 2001.
© Springer-Verlag Berlin Heidelberg 2001

time. In addition, when continuous image frames are processed, temporal parallelism can be exploited to improve computing performance even further. That is, algorithms at each level can be mapped to a processing layer (or cluster) of a multi-tasking architecture and executed simultaneously to create a pipelining effect. In the remainder of this paper, as well as in our problem definition, we will name a single pipeline, made of multiprocessing tasks (*MPT*), as a job.

In general, high performance parallel computing requires two techniques: program partitioning and task scheduling. Program partitioning deals with finding best grain size for the parallel algorithm considering the trade-off between parallelism and overhead. There are many techniques introduced in literature including simple heuristics, graph partitioning techniques, as well as meta-heuristics [1,4]. The main approach in these studies is to partition a task into subtasks considering network topology, processor, link, memory parameters and processor load balance to optimize the performance of computation. On the other hand, task scheduling deals with optimally scheduling *MPT*s so that overall makespan of the parallel application can be minimized. Various aspects of task scheduling have been studied in literature including deterministic and dynamic tasks, periodic tasks, preemptive, and non-preemptive tasks [1,2,4].

In different to these studies, we focus on job scheduling problem. As mentioned a job consists of multiple interdependent *MPT*s. The job scheduling problem is basically finding a sequence of jobs that can be processed on the system in minimum time. This problem as it stands is very complex; therefore we study a more restricted case in terms of computing platform and job parameters. In this paper, we consider jobs with deterministic parameters processed on a multi-tasking architecture with only two layers (or clusters). In our earlier study, we have developed list based heuristic algorithms especially for dynamic scheduling of jobs [6]. In the dynamic case, once a schedule is obtained it is implemented by control processors of the physical system. Most of the multi-tasking architectures employ a master-slave organization at each independent layer where master processor is responsible for initiating the processes. These heuristics provided fast solutions though their minimization of makespan were limited. On the other hand, for the deterministic cases scheduling can be done off-line during program compilation stage. This allows to employ more complex local search algorithms such as simulated annealing, tabu search, and genetic algorithms. These algorithms typically search for improved solutions until a stopping criterion is reached. It is most likely to find a better solution with these algorithms though their execution time is long due to their iterative nature. In this paper, we study simulated annealing and tabu search algorithms and evaluate their performances. In the following, a formal definition of the problem, simulated annealing and tabu search algorithms, and computational studies will be presented.

2 Basic Parameters and Problem Definition

We consider a set J of n independent and simultaneously available jobs to be processed in a computing platform with two multiprocessor layers where layer j has m_j

identical parallel processors, $j=1,2$. The level of pipeline in each job is the same and compatible with the number of processing layers available in the computing platform. Each job J_i J has two multiprocessor tasks (*MPT*s), namely $(i,1)$ and $(i,2)$. $MPT(i,j)$ should be processed on $size_{ij}$ number of processors simultaneously at layer j for a period of p_{ij} without interruption ($i=1,2,...,n$ and $j=1,2$). Hence, each $MPT(i,j)$ is characterized by its processing time, p_{ij}, and its processor requirement, $size_{ij}$ ($i=1,2,...,n$ and $j=1,2$). All the processors are continuously available from time 0 onwards and each processor can handle no more than one *MPT* at a time. Jobs flow through from layer 1 to layer 2 by utilizing any of the processors and by satisfying the *MPT* constraints. The objective is to find an optimal schedule for the jobs so as to minimize the maximum completion time of all jobs, i.e. the makespan, $C_{\max}$.

As in most allocation methods, we assume that processors are capable of simultaneously executing a task and performing a communication. This assumption is also based on the practical fact that majority of novel parallel architectures possess such feature. In our computations, communication cost between the subtasks is considered, though, for the sake of simplicity, this cost is included in p_{ij} as part of the total time that processors are occupied while performing a task.

3 Task Mapping Heuristic

Task mapping heuristic allocates tasks from a given job list by simply evaluating processor availability of the underlying hardware and requirements of *MPT*s. The algorithm performs following steps:

Step 1. Given a sequence S of the jobs, construct a schedule in layer 1 by assigning the first unassigned $MPT(i,1)$ of job J_i in S to the earliest time slot where at least $size_{i1}$ processors are available.

Step 2. As the *MPT*s are processed and finished in layer 1 in order, their counterpart became available to be processed in layer 2. Hence, schedule available *MPT*s to the earliest time slot in layer 2 by also taking into account their sequence in S.

4 Simulated Annealing

The stochastic methodologies can be used to improve the quality of allocations. Simulated annealing [9], *SA*, is an example to such methods. It performs heuristic hill climbing to transverse a search space in a manner, which is resistant to stopping prematurely at local critical points that are less optimum than the global one. As it is

known, in order to achieve this, *SA* scheme moves from one solution to another with the probability defined by the following equation:

$p(n) = \exp \quad \frac{E}{T(n)}$ where E is the difference in cost between the current solution and the new solution and $T(n)$ is a control parameter, which is also called 'temperature' at step n. A new state is accepted whenever its cost, or energy function, is better than the one associated with the previously accepted state. T is analogous to temperature associated with physical processes of the annealing. In general, T, is initialized with the value T_{init} and is then decreased in the manner dictated by the associated cooling schedule until it reaches the freezing temperature.

In order to apply *SA* to a practical problem several decisions have to be made. Next, we present our approach for each of these decisions.

Initial Solution: The initial solution is generated by setting all jobs in ascending order of job indices.

Neighborhood generation mechanism: A neighbor of the current solution is obtained in various ways. One method is to exchange two randomly chosen jobs from the priority list. This method is called *interchange neighborhood*. A special case of interchange neighborhood is simple *switch neighborhood*. It is defined by exchanging a randomly chosen job with its predecessor. Third method is called *shift neighborhood*, which involves removing a randomly selected job from one position in the priority list and putting it into another randomly chosen position. We have employed a preliminary computational experiment to examine the performance of these three methods. The results showed that the best performing neighborhood generation mechanism is *interchange* method. It is followed by *shift* and *simple switch* methods. Hence, *interchange* method is employed in our further experiments.

Objective function: The value of the objective function is defined as minimal value obtained for the completion time of all jobs, i.e. the makespan, $C_{\max}$.

Cooling Strategy: A simple cooling strategy is employed in our implementation. Temperature is decreased in an exponential manner with $T_i = \quad T_{i\ 1}$ where $\quad < 1$. In our implementation, value is selected as 0.998 after repetitive experiments.

Initial Temperature: It is important to select an initial temperature high enough to allow a large number of probabilistic acceptances. The initial value of temperature is selected using the formula: $T_o = \frac{\overline{E}_{avg}}{\ln(x_0)}$. Here $\overline{E}_{avg}$ is the average increase in the cost for a number of random transitions. Initial acceptance ratio, x_0, is defined as the number of accepted transitions divided by the number of proposed transitions. These parameters estimated after 50 randomly permuted neighborhood solution of the initial solution.

Stopping criterion: We have employed two stopping rules simultaneously. The first rule is the fixed number of iterations. The second rule compares average performance

deviation of the solution from the lower bounds and if it is less then 1% procedure is ended.

5 Tabu Search

Tabu search, *TS*, is another local search method, which is guided by the use of adaptive memory structures [7]. This method has been successfully applied to obtain optimal or sub-optimal solutions to optimization problems. The basic idea of the method is to explore the solution space by a sequence of moves made from one solution to another solution. However, to escape from locally optimal solutions and to prevent cycling, some moves are classified as forbidden or tabu. In the basic short term strategy of *TS*, if there is no better solution found than the current one, s_n, a move to the best possible solution, s, in the neighborhood $N(s_n)$ (or a sub-neighborhood $N'(s_n)$ $N(s_n)$ in the case $N(s_n)$ is too large to be explored efficiently) is performed. A certain number of the last visited solutions are stored in tabu list such that if a solution s is already in the list, the move from current solution (s_n s) is prohibited.

One of the main decision areas of *TS* is specification of a neighborhood structure and possibly of a sub-neighborhood structure. The three neighborhood generation strategies, discussed earlier in *SA* section, are also experimented with *TS* and interchange strategy is found to be the most effective one. For the sub-neighborhood $N'(s_n)$, we pick at random a fixed number of solutions in $N(s_n)$.

In the tabu list, we keep a fixed number of last visited solutions. We have also experimented keeping track of moves made instead of the solution sets. In this case, the computation time was shorter though we did not observe any significant advantage over the solution provided. We have experimented two methods for updating tabu list. These are the elimination of the farthest solution stored in the list, and removing the worst performing solution from the list. For the second method, an additional list to keep makespan values of the solutions in tabu list is required since the performance of a solution is measured with the makespan. This method resulted in slightly better performance than the first one. However, the tactical choices to improve the efficiency of the *TS* algorithm are somewhat longer than the *SA* and for this problem case performance of *TS* algorithm with the standard choices were slightly behind the *SA*.

6 Computational Experiments

Our computational study aims to analyze performance of the *SA* and *TS* methods on the minimization of makespan, as well as to investigate the effect of task characteris-

tics, and processor configurations on the performance. We consider different processing time ratios and different processor configurations for the randomly generated problems as explained below. In order to make sure a comparable computational effort committed by each heuristic, the stopping criterion for the following experiments defined as a fix number of solutions visited. This number has been set at 5000. We also compared these results with our earlier study where we have analyzed performance of several list-based heuristics for the job-scheduling problem.

The number of jobs was selected as $n = 10,30,50$. We have selected following two processing time ratios as defined in [10]. These are a) $p_{i1} \sim U[1,40]$ and $p_{i2} \sim U[1,40]$ b) $p_{i1} \sim U[1,40]$ and $p_{i2} \sim U[1,20]$ $(i = 1,2,...,n)$. The number of processors of multi-layer system was chosen according to following two configurations: a) More processors at layer 1, $m_1 = 2m_2 = 2^k$; b) Identical number of processors at each layer, $m_1 = m_2$; where $k = 1,2,3.$

For every $MPT(i, j)$, an integer processor requirement at layer j was generated from a uniform distribution over $[1, m_j]$ ($i = 1,2,...,n$ and $j = 1,2$). For each combination of processing time ratio and processor configuration of the architecture 25 problems were generated which are used to test the performance of *SA* and *TS* algorithms. In this section, we present the results of our computational study. For comparison, we have also included the performance of four best performing priority based heuristic algorithms from our earlier study where we have experimented 48 different heuristics that are a combination of 24 sequencing rules and two task mapping heuristics. The first heuristic algorithm, *H1*, obtains a sequence of jobs by applying Johnson's algorithm, *JA*, [8] assuming that $size_{ij} = m_j = 1$ ($i = 1,2,...,n$ and $j = 1,2$). Whereas, in the second heuristic algorithm, *H2*, a sequence of jobs obtained by first sorting tasks in non-increasing order of layer 2 processor requirements and then by sorting each group of tasks requiring same number of processors in non-increasing order of their layer 2 processing times. The sequencing rule in the third algorithm, *H3*, obtains a job list by simply sorting tasks in non-increasing order of layer 2 processing times. In heuristic *H4*, a set of job sequence is obtained by sorting the tasks in non-increasing order of $p_{i1}size_{i1} + p_{i2}size_{i2}$. In addition, we have also included the result of a heuristic based on random selection of jobs. All the algorithms are implemented using C++ and run on a PC with 350 Mhz Pentium II processor. Results are presented in terms of Average Percentage Deviation (APD) of the solution from the lower bound. The percentage deviation is defined as $((C_{\max}(HE) \quad LB)/LB) \quad 100$ where $C_{\max}(HE)$ denotes the $C_{\max}$ obtained by heuristic algorithms, that is SA, TS or list based heuristics. LB indicates the minimum of five lower bounds used [6]. The APD of each solution are presented in Figures 1 and 2.

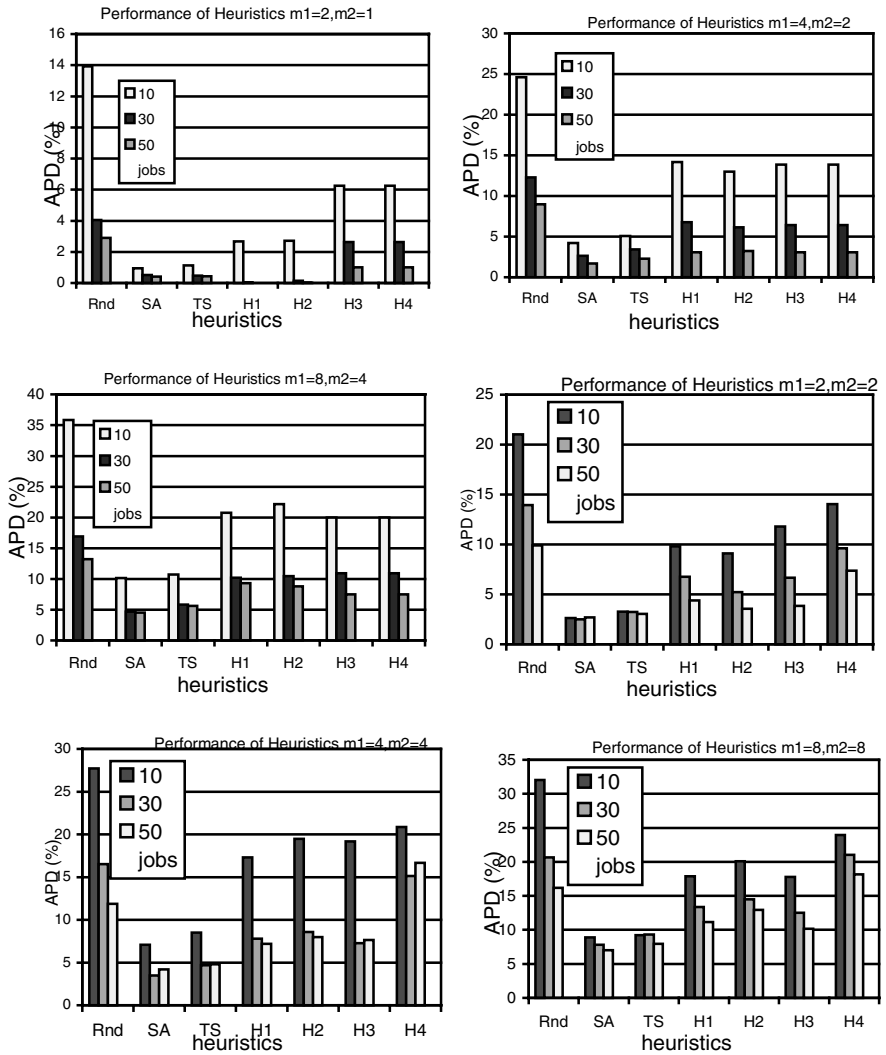


Fig. 1. Average percentage deviation of each algorithm for *P1*:*P2*=40:40.

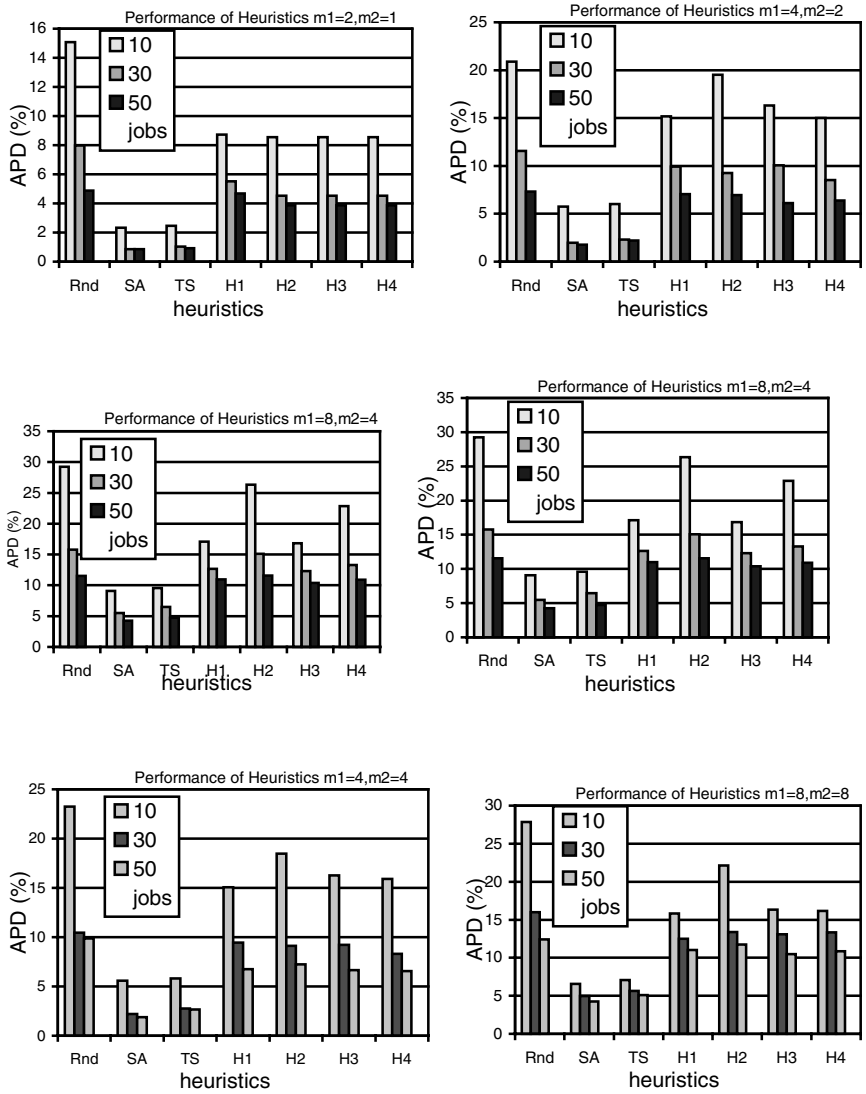


Fig. 2. Average percentage deviation of each algorithm for *P1*:*P2*=40:20.

The computational study shows that in all the cases *SA* and *TS* significantly outperform random sampling heuristic. In all the experiments, these metaheuristics delivered a better solution than random sampling ranging as high as 81 percent and as minimum as 14.5 percent. In none of the experiments, random selection encounters a solution at lower bounds or closer. The makespan minimization achieved by both SA and TS are quite similar, though, in most of the cases SA delivers a better result. In most of the

cases, SA converges to a reasonable solution within 500 iterations while TS converges within 1000 iterations.

It is also observed that in general the APD results of algorithms are better for the processing time ratio of 40:20 than the ratio 40:40. This can be explained as having a larger range for the main characteristic of the problem makes it difficult to schedule tasks, as it is more likely to have unbalanced processor loads. From Figures 1 and 2, APD seems to decrease as the number of jobs increases for each heuristic algorithm. This is explained as the number of job increases lower bound becomes more effective and close to the optimal solution. APD also deteriorates with the increasing number of processors. In addition, with increasing layer 2 to layer 1 processor ratio APD deteriorates. As layer 2 processors dictates the completion time of jobs, the increase of number of processors at this layer also increases the possibility of having idle processors, which consequently reduces the efficiency.

7 Summary

In this paper, a job-scheduling problem on a multi-tasking multiprocessor environment is considered. A job is made of interrelated multiprocessor tasks, which are modeled with their processing requirements and processing times. Two metaheuristic algorithms have been applied for the solution and their performance have been evaluated based on their capacity to minimize makespan. We compared these results with our earlier study where we have developed heuristic algorithms using simple sequencing rules. The results showed that metaheuristics significantly outperformed the list based heuristics. However, due to their large computation times they can be used in deterministic cases. So far, we have considered restricted case of the problem; a more general case will be dealt in our further study.

References

1. Błażewicz J., Ecker K. H., Pesch E., Schmidt G. and Weglarz J., Scheduling Computer and Manufacturing Processes, Springer-Verlag, Berlin, 1996
2. Bokhari S. H., Assignment Problems in Parallel and Distributed Computing, Kluwer Academic, Boston (1987)
3. Choudhary A. N., Patel J. H and Ahuja N., NETRA: A Hierarchical and Partitionable Architecture for Computer Vision Systems, IEEE Trans. Parallel and Distributed Systems, Vol. 4, (1996) 1092-1104
4. El-Revini H., partitioning and and scheduling, in:A.Y.H. Zomaya (ed.), Parallel and Distrib-uted Computing Handbook, McGraw-Hill, New York, (1996) 239-273

5. Ercan M. F. and Y. F. Fung, Real-time Image Interpretation on a Multi-layer Architecture, IEEE TENCON'99, Vol. 2, (1999) 1303-1306.
6. Ercan M.F., C. Oguz and Y. F. Fung, Scheduling Image Processing Tasks in A Multi-layer System, in print Computers and Electrical Engineering.
7. Glover F. and Laguna, Tabu search, Kluwer Academic Publishers, Boston, (1997)
8. Johnson S. M., Optimal Two and Three-stage Production Schedules with Setup Times Included, Naval Research Logistic Quarterly, Vol. 1, (1954) 61-68
9. Kirkpatrick S., Gelatt, C. D., and Vecchi M. P., Optimization by Simulated Annealing, Science, Vol. 220, (1983) 671-680
10. Lee C.Y and Vairaktarakis G.L., Minimizing Make Span in Hybrid Flow-Shops, Operations Research Letters, Vol. 16, (1994) 149-158
11. Scala M. L., Bose A., Tylavsky J., and Chai J.S., A Highly Parallel Method for Transient Stability Analysis, IEEE Transactions on Power Systems, Vol. 5, (1990) 1439-1446
12. Siegel H.J., Siegel L.J., Kemmerer F.C., Mueller P.T., Smalley H.E., and Smith S.D., PASM- A Partitionable SIMD/MIMD System for Image Processing and Pattern Recognition, IEEE Trans. Computers, Vol. C-30, (1981) 934-947
13. Weems C.C., Riseman E.M. and Hanson A.R., Image Understanding Architecture: Exploiting Potential Parallelism in Machine Vision, IEEE Computer, Vol. 25 (1992) 65-68

Automatic Performance Tuning in the UHFFT Library

Dragan Mirković[1] and S. Lennart Johnsson[1]

Department of Computer Science
University of Houston
Houston, TX 77204
mirkovic@cs.uh.edu, johnsson@cs.uh.edu

Abstract. In this paper we describe the architecture–specific automatic performance tuning implemented in the UHFFT library. The UHFFT library is an adaptive and portable software library for fast Fourier transforms (FFT).

1 Introduction

The fast Fourier transform (FFT) is one of the most popular algorithms in science and technology. Its uses range from digital signal processing and data compression to numerical solution of partial differential equations. The importance of the FFT in many applications has provided a strong motivation for development of highly optimized implementations. The growing complexity of modern microprocessor architectures with multi-level memory hierarchies and instruction level parallelism has made performance tuning increasingly difficult. In particular, FFT algorithms have a number of inherent properties that make them very sensitive to the memory hierarchy mapping. These include the recursive structure of the FFT algorithm, its relatively high efficiency ($O(n \log n)$) which implies a low floating–point v.s. load/store instruction ratio, and the strided data access pattern. Besides that, the unbalance in the number of additions and multiplications reduces some of the advantages of modern superscalar architectures.

The need for the FFT codes has forced many application programmers to manually restructure and tune their codes. This is a tedious and error prone task, and it requires the expertise in computer architecture. The resulting code is less readable, difficult to maintain and not easily portable.

One way to overcome these difficulties is to design codes that adapt themselves to the computer architecture by using a dynamic construction of the FFT algorithm. The adaptability is accomplished by using a library of composable blocks of code, each computing a part of the transform, and by selecting the optimal combination of these blocks at runtime. A very successful example that uses this approach is the FFTW library [1]. We have adopted similar approach in the UHFFT library and extended it with a more elaborate installation tuning and richer algorithm space for execution.

For this approach to be efficient the blocks of code (*codelets* in FFTW lingo) in the library should be highly optimized and tuned to the specific architecture

V.N. Alexandrov et al. (Eds.): ICCS 2001, LNCS 2073, pp. 71–80, 2001.
© Springer-Verlag Berlin Heidelberg 2001

and the initialization should be fast and inexpensive. These goals can be achieved by performing the time consuming tasks required by the optimization during the installation of the library. In UHFFT, we first use a special–purpose compiler to generate and tune the codelets in the library. Second, we do most of the time consuming performance measurements during the installation of the library. The major novelty in the UHFFT library is that most of the code is automatically generated in the course of the installation of the library with an attempt to tune the installation to the particular architecture. To our knowledge this is the only FFT library with such capability. Although several other public domain libraries make use of automatic code generation techniques similar to ours, their code is usually pregenerated and fixed for all platforms. Even if they allow for possible modifications of the generated code, this modifications are cumbersome and not at all automatic. On the other hand our code is generated and optimized at the time of installation.

In this paper we give an overview of the automatic performance tuning techniques incorporated in the UHFFT library. The rest of the paper is organized as follows. Section 2 gives the basic mathematical background for the *polyalgorithmic* approach used both to build the library of codelets and to combine them during the execution. Section 3 describes the automatic optimization and tuning methodology used in the UHFFT.

2 Mathematical Background

The Fast Fourier Transform (FFT) is a method used for the fast evaluation of the Discrete Fourier Transform (DFT). The DFT is a matrix–vector product that requires $O(n^2)$ arithmetic operations to compute. Using the FFT to evaluate the DFT reduces the number of operations required to $O(n \log n)$. In this chapter we give a short list of the algorithms used in the UHFFT library. We refer the reader to [2], [3], and [4] for the more detailed description of the algorithms. In particular, the notation we use here mostly coincides with the notation in [2].

Let $\mathbf{C}^n$ denote the vector space of complex *n–vectors* with components indexed from zero to $n-1$. The Discrete Fourier Transform (DFT) of $\mathbf{x} \in \mathbf{C}^n$ is defined by

$$\mathbf{y} = W_n \mathbf{x} \tag{1}$$

where $W_n = (w_{lk})_{l,k=0}^{n-1}$ is the DFT matrix with elements $w_{lk} = \omega_n^{lk}$, and $\omega_n = e^{-2\pi i/n}$ is the principal nth root of unity. The fast evaluation is obtained through factorization of W_n into the product of $O(\log n)$ sparse matrix factors so that (1) can be evaluated as

$$W_n \mathbf{x} = (A_1 A_2 \dots A_r)\mathbf{x} \tag{2}$$

where matrices A_i are sparse and $A_i \mathbf{x}$ involves $O(n)$ arithmetic operations. The factorization (2) for given n is not unique, and possible variations may have properties that are substantially different. For example, it can be shown that when $n = rq$, W_n can be written as

$$W_n = (W_r \otimes I_q) D_{r,q} (I_r \otimes W_q) \Pi_{n,r}, \tag{3}$$

where $D_{r,q}$ is a diagonal *twiddle–factor* matrix, $D_{r,q} = \bigoplus_{k=0}^{r-1} \Omega_{n,q}^k$, $\Omega_{n,q} = \bigoplus_{k=0}^{q-1} \omega_n^k$, and $\Pi_{n,r}$ is a mod–r sort permutation matrix. The algorithm (3) is the well known Cooley–Tukey [5] mixed–radix splitting algorithm. In this algorithm a non–trivial fraction of the computational work is associated with the construction and the application of the diagonal scaling matrix $D_{r,q}$. The *prime factor* FFT algorithm (PFA) [6,7,8] removes the need for this scaling when r and q are relatively prime, i.e., $\gcd(r, q) = 1$. This algorithm is based upon splittings of the form:

$$W_n = P_1(W_r \otimes I_q)(I_r \otimes W_q)P_2 = P_1(W_r \otimes W_q)P_2 = P^T(W_r^{(\alpha)} \otimes W_q^{(\beta)})P, \quad (4)$$

where P_1, P_2 and P are permutations and $W_r^{(\alpha)} = (w_{lk}^{(\alpha)})_{l,k=0}^{n-1}$ is the *rotated* DFT matrix with elements $w_{lk}^{(\alpha)} = \omega_n^{\alpha lk}$.

If q is not a prime number the above algorithms can be applied recursively, and this is the heart of the fast Fourier transform idea. In some cases the splitting stages can be combined together and, with some simplifications, the result may be a more efficient algorithm. The well known example is the *split–radix* algorithm proposed by Duhamel and Hollmann [9], which can be used when n is divisible by 4. Assume that $n = 2q = 4p$ and let $\mathbf{x} \in \mathbf{C}^n$. By using (3) with $r = 2$ we obtain

$$W_n = (W_2 \otimes I_q)D_{n,q}(I_2 \otimes W_q)\Pi_{n,2}. \quad (5)$$

The *split–radix* algorithm is obtained by using the same formula again on the second block of the block–diagonal matrix $I_2 \otimes W_q = W_q \oplus W_q$, and rearranging the terms such that the final factorization is of the form

$$W_n = B(W_q \oplus W_p \oplus W_p)\Pi_{n,q,2}. \quad (6)$$

Here, B is the split–radix butterfly matrix and $\Pi_{n,q,2}$ is the split–radix permutation matrix, $\Pi_{n,q,2} = (I_q \oplus \Pi_{q,2})\Pi_{n,2}$. The efficiency of the split–radix algorithm follows from simplifications of the butterfly matrix

$$B = (W_2 \otimes I_q)D_{n,q}[I_q \oplus (W_2 \otimes I_p)D_{q,p}], \quad (7)$$

which, after some manipulations, can be written as

$$\begin{aligned} B &= (W_2 \otimes I_q)[I_q \oplus (SW_2 \otimes I_p)](I_q \oplus \Omega_{n,p} \oplus \Omega_{n,p}^3) \\ &= B_a B_m, \end{aligned} \quad (8)$$

where $S = 1 \oplus -i$; $B_a = (W_2 \otimes I_q)[I_q \oplus (SW_2 \otimes I_p)]$ is the additive and $B_m = (I_q \oplus \Omega_{n,p} \oplus \Omega_{n,p}^3)$ is the multiplicative part of the butterfly matrix B.

When n is a prime, there is a factorization of W_n proposed by Rader [10,3, 2] involving a number–theoretic permutation of W_n that produces a circulant or a skew–circulant submatrix of order $n - 1$. The indexing set $\{0, \ldots, n-1\}$ for prime n is a field with respect to addition and multiplication modulo n, and all of its nonzero elements can be generated as powers of a single element called a *primitive root*. The permutation induced by the powers of the primitive root r

$$\mathbf{z} = Q_{n,r}\mathbf{x}, \quad z_k = \begin{cases} x_k & if \quad k = 0, 1 \\ x_{<r^{k-1}>_n} & if \quad 2 \le k \le n-1 \end{cases} \quad (9)$$

is called the *exponential permutation* associated with r. It can be shown that

$$W_n = Q_{n,r}^T \begin{pmatrix} 1 & \mathbf{1}^T \\ \mathbf{1} & C_{n-1} \end{pmatrix} Q_{n,r^{-1}} = Q_{n,r}^T \begin{pmatrix} 1 & \mathbf{1}^T \\ \mathbf{1} & S_{n-1} \end{pmatrix} Q_{n,r}, \tag{10}$$

where $\mathbf{1}$ is a vector of all ones and C_{n-1} and S_{n-1} are circulant and skew–circulant matrices respectively, generated by the vector $\mathbf{c} = (\omega_n, \omega_n^r, \ldots, \omega_n^{r^{n-2}})^T$. Both C_{n-1} and S_{n-1} can be diagonalized by W_{n-1},

$$C_m = W_m^{-1}\mathrm{diag}(W_m\mathbf{c})W_m \quad \text{and} \quad S_m = W_m\mathrm{diag}(W_m^{-1}\mathbf{c})W_m,\ m = n-1. \tag{11}$$

This effectively reduces the prime size problem to a non–prime size one for which we may use any other known algorithm. Asymptotically optimal algorithms for prime n can be obtained through full diagonalization of C_{n-1} and S_{n-1}. When n is small, though, a partial diagonalization with $(W_2 \otimes I_{(n-1)/2})$ may result in a more efficient algorithm [3].

The list of possible FFT algorithms by no means ends with the four basic algorithms described in this section, but these four algorithms and their variants are used as the basic building blocks in the UHFFT library.

3 Performance Tuning Methodology

The optimization in the UHFFT library is performed on two levels and a coarse flowchart of the performance tuning methodology is shown in Figure 1. The first (high) level optimization consists of selecting the optimal factorization of the FFT of a given size, into a number of factors, smaller in size, for which an efficient DFT codelet exists in our library. The optimization on this level is performed during the initialization phase of the procedure, which makes the code adaptive to the architecture it is running on.

The second (low) level optimization involves generating a library of efficient, small size DFT codelets. Since the efficiency of the code depends strongly on the efficiency of the codelets themselves, it is important to have the best possible performance for the codelets to be able to build an efficient library. The code generation and the architecture specific tuning is a time consuming process and it is performed during the installation of the library. We have a small number of installation options that can be specified by the user. At this moment these options are restricted to the range of sizes and dimensions for which the library should be optimized. We are planning to extend the range of options to include the interface, data distribution and parallelization methods in the future releases. We may also include some application specific options like known symmetries in the data, restrictions on the size of the library and the memory used by the code. The extent of the additional options will strongly depend on the feedback we get from the users. The idea is to exploit the flexibility of code generation and optimization tools which are built in the library for the benefit of the user and to allow for a significant and simple customization of the library.

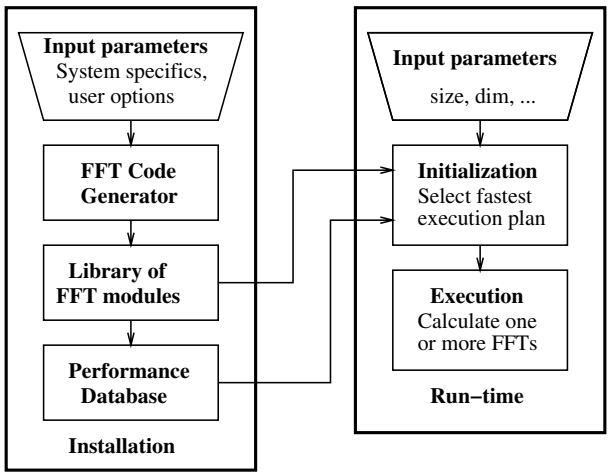


Fig. 1. Performance tuning methodology in UHFFT.

3.1 Execution Plan Generation

Given the parameters of the problem, the initialization routine selects the strategy in terms of execution time on the given architecture. This selection involves two steps. First, we use a combination of the mixed–radix, split–radix and prime factor algorithm splittings to generate a large number of possible factorizations for a given transform size. Next, the initialization routine attempts to select a strategy that minimizes the execution time on the given architecture.

The basis for generating execution plans are the library of codelets and two databases: the *codelet database* storing information about codelet execution times, and the *transform database* that stores information about the execution times for entire transforms. The codelet database is initialized during installation of the library as a part of the benchmarking routine.

The transform database stores the best execution plan for different size transforms. The transform database is initialized for some of the popular FFT sizes during installation (such as power of 2 and PFA sizes).

For transform sizes that are not in the database, an execution plan must be created and this can be done in two different ways.

The first method is to empirically find the execution plan that minimizes the execution time by executing all possible plans for the given size, and choose the plan with the best performance. This method ensures that the plan selected will indeed result in the smallest execution time for all choices possible within the UHFFT library, but the time required to find the execution plan may be quite large for large size FFTs. So, unless many transforms of a particular size are needed this method is not practical.

The second method is based on estimating the performance of different execution plans using the information in the codelet database. For each execution plan feasible with the codelets in the library the expected execution time is

derived based on the codelets being used in the plan, the number of calls to each codelet, and the codelet performance data in the codelet database. The estimation algorithm also takes into account the input and output strides and transform direction (forward or inverse). It also accounts for the twiddle factor multiplications for each plan as the number of such multiplications depend on the execution plan.

For large transform sizes with many factorizations, the estimation method is considerably faster than the empirical method. The quality of the execution plan based on the estimation approach clearly relies heavily on the assumption that codelet timings can be used to predict transform execution times, and that the memory system will have a comparable impact on all execution plans.

The adaptive approach used by UHFFT is very similar to the one used by the FFTW library [1]. The main difference is the set of algorithms used generate the collection of possible execution strategies. FFTW uses the mixed–radix and Rader's algorithm while we currently use mixed–radix, split–radix and prime factor algorithm. While we are still planning to include the Rader's algorithm, its significance at the execution level is to have an asymptotically optimal code for all transform sizes (including the prime sizes and sizes containing prime factors not included in the library of codelets). The performance that can be achieved for these sizes, though, is relatively low when compared to the neighboring (non–prime) transform sizes. For example the transforms for sizes 32 and 128 are approximately ten times faster then the transforms for sizes 31 and 127 respectively. On the other hand, both the split–radix and the prime factor algorithm provide for the richer and more efficient algorithm space covered by the library. We illustrate that by comparing the performance of UHFFT versus FFTW for the PFA transform sizes in Figure 2. Here UHFFT uses the PFA to combine the codelets, while FFTW uses the mixed–radix algorithm.

3.2 Library of FFT Modules

The UHFFT library contains a number of composable blocks of code, called *codelets*, each computing a part of the transform. The overall efficiency of the code depends strongly on the efficiency of these codelets. Therefore, it is essential to have a highly optimized set of DFT codelets in the library. We divide the codelet optimization into a number of levels. The first level optimization involves reduction of the number of arithmetic operations for each DFT codelet. The next level of optimization involves the memory hierarchy. In current processor architectures, memory access time is of prime concern for performance. Optimizations involving memory accesses are architecture dependent and are performed only once during the installation of the library.

The codelets in our library are generated using a special purpose compiler that we have developed. The FFT code generation techniques have been in use for more than twenty years (see an overview given by Matteo Frigo in [11]). Most of the existing implementations are restricted to complex transforms with a predetermined generation algorithm. A notable exception is the FFTW generator *genfft*, which not only uses a flexible set of algorithms, but also deals

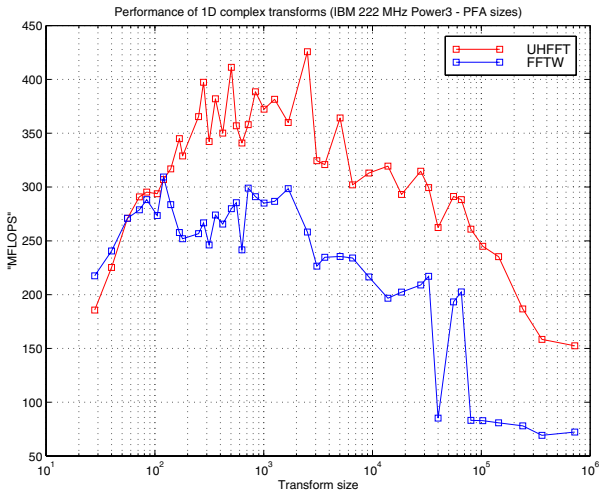


Fig. 2. Graph of the performance of UHFFT versus FFTW on a 222 MHz IBM Power 3 processor for selected transform sizes that can be factored into mutually prime powers of 2, 3, 5, 7, 11, and 13. The peak performance of 409 MFLOPS achieved by the UHFFT PFA plan for $n = 2520$ is not only higher than the FFTW performance for the same size (258 MFLOPS), it is also higher than the performance of FFTW for any size we tested on this processor. The peak performance achived by FFTW was 397 MFLOPS for $n = 64$.

with optimization and scheduling problems in a very efficient way. The FFTW code generator is written in Objective Caml [12], a powerful and versatile dialect of the ML functional language. Although the Caml capabilities simplify the construction of the code generator, we find the dependence on a large and nonstandard library an impediment in the automatic tuning of the installation. For that reason we have decided to write the UHFFT code generator in C. This approach makes the code generation fast and efficient and the whole library is more compact and ultimately portable. We have also built the enough infrastructure in the UHFFT code generator to match most of the functionalities of *genfft*. Moreover, we have added a number of derived data types and functions that simplify the implementation of standard FFT algorithms. For example, here is a function that implements the mixed–radix algorithm (3).

```
/*
*       FFTMixedRadix()       Mixed-radix splitting.
*       Input:
*               r       radix,
*               dir     direction of the transform,
*               rot     rotation of the transform,
*               u       input expression vector.
*/
ExprVec *FFTMixedRadix(int r, int dir, int rot, ExprVec *u)
```

```
{
        int     m, n = u->n, *p;

        m = n/r;
        p = ModRSortPermutation(n, r);
        u = FFTxI(r, m, dir, rot,
                TwiddleMult(r, m, dir, rot,
                IxFFT(r, m, dir, rot, PermuteExprVec(u, p))));
        free(p);
        return u;
}
```

The functions `FFTxI()` and `IxFFT` correspond to the expressions $(W_r \otimes I_m)$ and $(I_r \otimes W_m)$ respectively, `TwiddleMult()` implements the multiplication with the matrix of twiddle factors $D_{r,q}$; the action of the mod-r sort permutation matrix $\Pi_{n,r}$ is obtained by calling the function `PermuteExprVec(u, p)`, where the permutation vector `p` is the output of the function `ModRSortPermutation(n, r)`.

The UHFFT code generator can produce DFT codelets for complex and real transforms of arbitrary size, direction (forward or inverse), and rotation (for PFA). It first generates an abstraction of the FFT algorithm by using a combination of Rader's algorithm, the mixed–radix algorithm, the split–radix algorithm and the PFA. The next step is the scheduling of the arithmetic operations such that memory accesses are minimized. We make effective use of temporary variables so that intermediate writes use the cache instead of writing directly to memory. We also use blocking techniques so that data residing in the cache is reused the maximum possible number of times without being written and re–read from main memory. Finally, the abstract code is unparsed to produce the desired C code. The output of the code–generator is then compiled to produce the executable version of the library. The structure of the library is given in Figure 3.

The performance depends strongly on the transform size, input and output strides and the structure of the memory hierarchy of a given processor. Once the executables for the library are ready, we benchmark the codelets to test its performance. These benchmark tests are conducted for various input and output strides of data. The results of these performance tests are then stored in a database that is used later by the execution plan generator algorithm during the initialization phase of an FFT computation. In Figure 4 we show a typical performance map on two different processors.

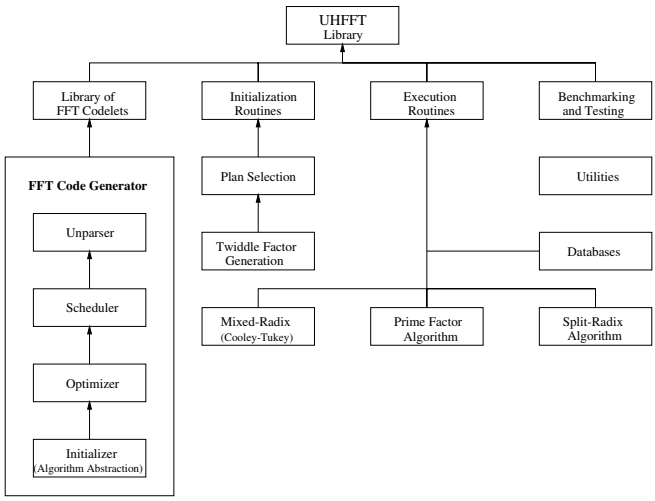


Fig. 3. UHFFT Library Organization.

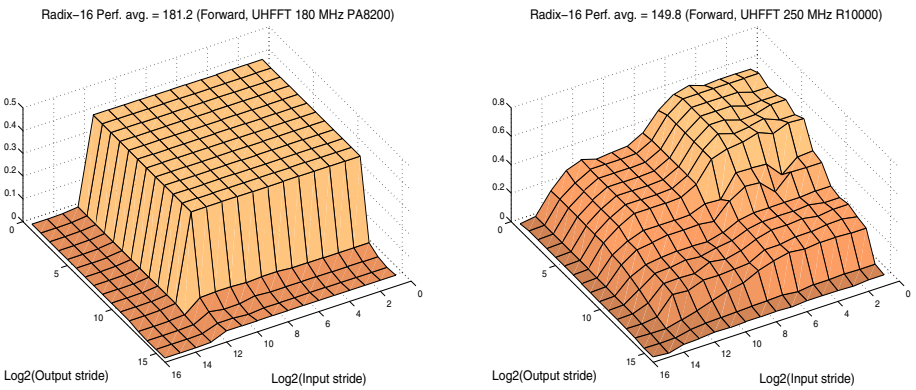


Fig. 4. Two examples of the size 16 codelet performance on 250 MHz SGI R10000 and 180 MHz HP PA8200. The SGI R10000 processor has peak performance of 500 MFlops. This processor has as primary caches a 32 KB two way set associative on chip instruction cache and a 32 KB two way set associative, two way interleaved on chip data cache with LRU replacement. It also has a 4 MB two way set associative L2 secondary cache per CPU. The SGI R10000 has 64 physical registers, each 64 bits wide. The HP PA8200 RISC microprocessor operating at 180 MHz is capable of 720 MFlops peak. It has a 1 MB direct mapped data cache and a 1 MB instruction cache. The performance is plotted as a fraction of the peak achivable performance. The di erence is typical for one versus two levels of cache.

References

[1] Matteo Frigo and Steven G. Johnson. The Fastest Fourier Transform in the West. Technical Report MIT-LCS-TR-728, MIT, 1997.
[2] Charles Van Loan. *Computational frameworks for the fast Fourier transform.* Philadelphia:SIAM, 1992.
[3] Richard Tolimieri, Myoung An, and Chao Lu. *Algorithms for Discrete Fourier Transforms and Convolution.* Springer–Verlag, New York, 1 edition, 1989.
[4] P. Duhamel and M. Vetterli. Fast Fourier Transforms: A Tutorial Review and a State of the Art. *Signal Processing*, 19:259–299, 1990.
[5] J.C. Cooley and J.W. Tukey. An algorithm for the machine computation of complex fourier series. *Math. Comp.*, 19:291–301, 1965.
[6] I.J. Good. The interaction algorithm and practical Fourier Analysis. *J. Royal Stat. Soc., Ser. B*, 20:361–375, 1958.
[7] L.H. Thomas. Using a computer to solve problems in physics. In *Application of Digital Computers.* Ginn and Co., Boston, Mass., 1963.
[8] C. Temperton. A Note on Prime Factor FFT Algorithms. *Journal of Computational Physics*, 52:198–204, 1983.
[9] P. Duhamel and H. Hollmann. Split Radix FFT Algorithms. *Electronic Letters*, 20:14–16, 1984.
[10] C. M. Rader. Discrete Fourier transforms when the number of data samples is prime. *Proceedings of the IEEE*, 56:1107–1108, 1968.
[11] Matteo Frigo. A Fast Fourier Transform Compiler. *Proceedings of the 1999 ACM SIGPLAN Conference on Programming Language Design and Implementation*, pages 169–180, 1999.
[12] Xavier Leroy. Le système Caml Special Light: modules et compilation efficace en Caml. Technical Report 2721, INRIA, November 1995.

A Modal Model of Memory*

Nick Mitchell[1], Larry Carter[2], and Jeanne Ferrante[3]

[1] IBM T.J. Watson Research Center
30 Saw Mill River Road
Hawthorne, NY 10532 USA
nickm@us.ibm.com
[2] University of California, San Diego and San Diego Supercomputing Center
carter@cs.ucsd.edu
[3] University of California, San Diego
ferrante@cs.ucsd.edu

Keywords: performance, model, cache, profiling, modal

* Contact author: Nick Mitchell, who was funded by an Intel Graduate Fellowship, 1999-2000. In addition this work was funded by NSF grant CCR-9808946. Equipment used in this research was supported in part by the UCSD Active Web Project, NSF Research Infrastructure Grant Number 9802219.

V.N. Alexandrov et al. (Eds.): ICCS 2001, LNCS 2073, pp. 81–96, 2001.
© Springer-Verlag Berlin Heidelberg 2001

Abstract. We consider the problem of automatically guiding program transformations for locality, despite incomplete information due to complicated program structures, changing target architectures, and lack of knowledge of the properties of the input data. Our system, the *modal model of memory*, uses limited static analysis and bounded runtime experimentation to produce performance formulas that can be used to make runtime locality transformation decisions. Static analysis is performed once per program to determine its memory reference properties, using *modes*, a small set of parameterized, kernel reference patterns. Once per architectural system, our system automatically performs a set of experiments to determine a family of kernel performance formulas. The system can use these kernel formulas to synthesize a performance formula for any program's mode tree. Finally, with program transformations represented as mappings between mode trees, the generated performance formulas can be used to guide transformation decisions.

1 Introduction

We consider the problem of automatically guiding program transformations despite incomplete information. Guidance requires an infrastructure that supports queries of the form, "under what circumstances should I apply this transformation?" [35,29,2,18,33]. Answering these queries in the face of complicated program structures, unknown target architecture, and lack of knowledge of the input data requires a combined compile-time/runtime solution. In this paper, we present our solution for automatically guiding locality transformations: the *modal model of memory*. Our system combines *limited* static analysis with *bounded* experimentation to take advantage of the *modal* nature of performance.

1.1 Limited Static Analysis

Many compilation strategies estimate the profitability of a transformation with a purely *static* analysis [10,30,28,25,11,14], which in many cases can lead to good optimization choices. However, by relying only on static information, the analysis can fail on two counts. First, the underlying mathematical tools, such as integer linear programming, often are restricted to simple program structures. For example, most static techniques cannot cope with indirect memory references patterns, such as $A[B[i]]$, except probabilistically [19]. The shortcomings of the probabilistic technique highlight the second failure of purely static strategies. Every purely static strategy must make *assumptions* about the environment in which a program will run. For example, [19] assumes that the B array is sufficiently long (but the SPECint95 `go` benchmark uses tuples on the order of 10 elements long [24]), whose elements are uniformly distributed (but the NAS integer sort benchmark [1] uses an almost-Gaussian distribution), and distributed over a sufficiently large range, r (for NAS EP benchmark, $r = 10$), and, even if r is known, might assume a threshold $r > t$ above which performs suffers (yet, t clearly depends on the target architecture). Our approach applies just enough static analysis to identify intrinsic memory reference patterns, represented by a tree of parameterized *modes*. Statically unknown mode parameters can be instantiated whenever their values become known.

1.2 Bounded Experimentation

Alternatively, a system can gather information via experimentation with candidate implementations. This, too, can be successful in many cases [4,3,34]. For example, in the case of tiling, it could determine the best tile size, given the profiled program's input data, on a given architecture [3,34]. However, such information is always relative to the particular input data, underlying architecture, and chosen implementation. A change to a different input, or a different program, or a new architecture, would require a new set of profiling runs. In our system, once per architecture, we perform just enough experimentation to determine the behavior of the system on our small set of parametrized modes. We can then use the resulting family of *kernel* performance formulas (together with information provided at run time) to estimate performance of a program implementation.

1.3 Modal Behavior

Our use of modes is based on the observation that, while performance of an application on a given architecture may be difficult to precisely determine, it is often modal in nature. For example, the execution time per iteration of a loop nest may be a small constant until the the size of cache is exceeded; at this point, the execution time may increase dramatically. The execution time of a loop can also vary with the pattern of memory references: a constant access in a loop may be kept in a register, and so be less costly than a fixed stride memory access. Fixed stride access, however, may exhibit spatial locality in cache, and so in turn be less costly than random access. Our approach, instead of modeling performance curves exactly, is to find the inflection points where performance changes dramatically on a given architecture.

Our system uses a specification of a small set of parametrized memory modes, as described in Section 2, as the basis for its analysis and experimentation. Each code analyzed is represented as a tree of modes (Section 3). Additionally, once per architecture, our system automatically performs experiments to determine the performance behavior of a small number of mode contexts (Section 4). The result is a set of kernel formulas. Now, given the mode tree representing a code, our system can automatically synthesize a performance formula from the kernel formulas. Finally, with transformations represented as mappings between mode trees, the formulas for the transformed and untransformed code can be instantiated at runtime, and a choice made between them (Section 5).

2 Reference Modes and Mode Trees

The central representation of our system is a *parameterized reference mode*. We introduce a set of three parameterized modes that can be combined into *mode trees*. Mode trees capture essential information about the memory reference pattern and locality behavior of a program. In this paper, we do not present the underlying formalism (based on attribute grammars); we refer the reader to [24].

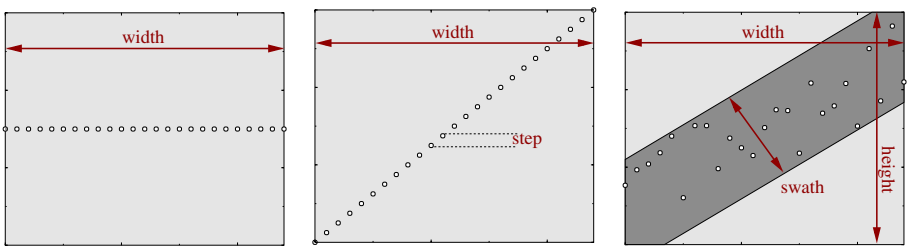
```
1  do i = 1, 25, 10
2      A(3)
3      B(i)
4      C(A(i))
5  end
```

Fig. 1. An example with three base references.

The central idea of locality modes is this: by inspecting a program's syntax, we can draw a picture of its memory reference pattern. While this static inspection may not determine the details of the picture *precisely* (perhaps we do not know the contents of an array, or the bounds of a loop), nevertheless it provides enough knowledge to allow the system to proceed.

2.1 Three Locality Reference Modes

Let's start with the example loop nest in Fig. 1, which contains three base array references. Each of the three lines in the body accesses memory differently. Line 2 generates the sequence of references $(3, 3, 3)$[1], which is a constant reference pattern. Line 3 generates the sequence $(1, 11, 21)$, a monotonic, fixed-stride pattern. Finally, we cannot determine which pattern line 4 generates precisely; it depends on the values stored in A. Yet we can observe, from the code alone, that the pattern has the possibility of being a non-fixed-stride pattern, unlike the other two patterns.



(a) κ mode (b) σ mode (c) ρ mode

Fig. 2. Visualizing the locality reference modes with memory reference patterns time flows horizontally, and space flows vertically. Each mode can be annotated with its parameters; for example, the σ mode has two parameters, step and width. More complicated patterns result from the composition of these three modes into mode trees.

Corresponding to these three patterns of references, we define three parameterized reference modes, denoted κ, σ, and ρ. They are visualized in Fig. 2.

[1] Here we use 3 as shorthand denoting the third location of the A array.

constant: κ represents a constant reference pattern, such as $(3, 3, 3)$. Visually, a constant pattern looks like a horizontal line, as shown in Fig. 2(a). This reference pattern has only one distinguishing (and possibly statically unknown) feature: the length of the tuple (i.e., *width* in the picture).

strided: σ represents a pattern which accesses memory with a fixed stride; e.g. $(1, 11, 21)$, as shown in Fig. 2(b). There are two distinguishing (and again, possibly unknown) parameters: the step or stride between successive reference, and the width.

non-monotonic: ρ represents a non-monotonic reference pattern: $(5, 12, 4)$. Visually, a non-monotonic pattern will be somewhere on a spectrum between a diagonal line and random noise; Figure 2(c) shows a case somewhere between these two extremes. A non-monotonic pattern has three possibly unknown parameters: the height, the width, and the point along the randomness spectrum, which we call *swath-width*.

2.2 Mode Trees Place Modes in a Larger Context

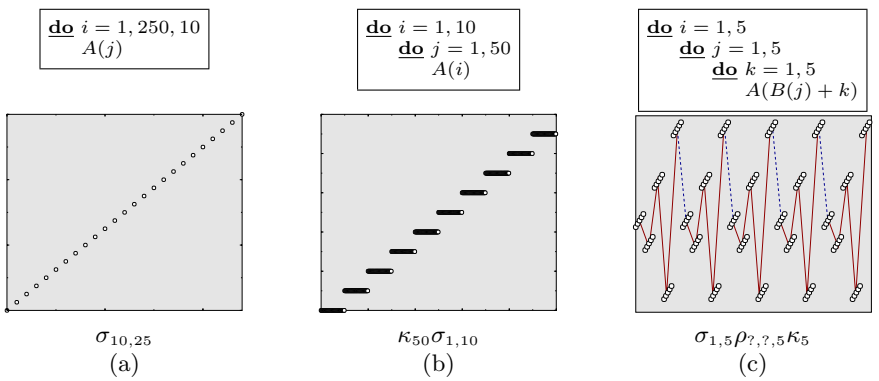


Fig. 3. Three example mode trees. Each example shows a loop nest, a reference pattern picture, and a tree of parameterized *modes*. We write these tree as a strings, so that $\kappa\sigma$ has κ as the child. Some of the parameters may not be statically known; we denote the value of such parameters by question marks (?).

Many interesting memory reference patterns cannot be described by a single mode. However, we can use a *tree* of modes for more complex situations. Consider the example given in Fig. 3(b): a doubly-nested loop with a single array reference, $A(i)$. With respect to the j loop, the memory locations being referenced do not change; this pattern is an instance of κ, the constant mode. With respect to the i loop, the memory locations fall into the σ mode. The reference pattern of this example is the *composition* of two modes: first κ (because j is the inner loop), then σ. If we draw this relationship as a tree, σ will be the parent of κ. Fig. 3(b)

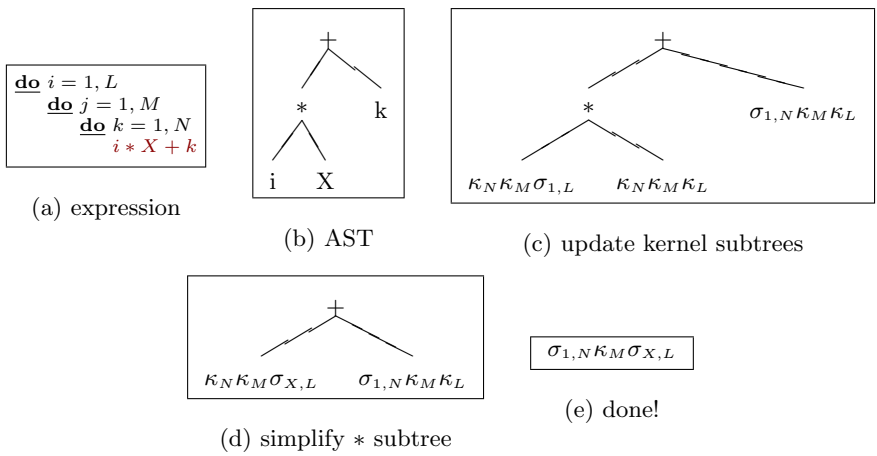
linearizes this tree to a string: $\kappa\sigma$, for notational cleanliness.[2] Figure 3(c) is an example of a triply-nested loop. Nested loops are instances of the parent-child mode relationship.

We can also have *sibling* mode relationships. This would arise for example in Fig. 1, where a single loop nest contains three array references. We do not discuss this case in this paper. In short, sibling relations require extending (and nominally complicating) the mode tree representation.

3 Lexical Analysis — from Codes to Modes

In Sec. 2, we introduced the language of modes: a mode is a class of abstract syntax trees, and a mode tree represents a mode in a larger context. We now briefly describe, mainly by example, how to instantiate a mode tree from a given abstract syntax tree.

First, identify some source expression of interest. For example, say we are interested in the expression $i*X+k$ in a three-deep loop nest, such as in Figure 4(a); say X is some loop-invariant subexpression. From this source expression, we can easily create its corresponding abstract syntax tree (AST), shown in Figure 4(b).



(a) expression (b) AST (c) update kernel subtrees

(d) simplify $*$ subtree (e) done!

Fig. 4. A lexical analysis example.

From the abstract syntax tree, we next update the "kernel" subtrees. Recall from Sec. 2, that a reference mode is a class of abstract syntax trees. A subtree is a *kernel* subtree of an AST if it belongs to the class of some reference mode [24]. For example, in the AST of Fig. 4(b), the mode σ validates the leaf nodes i and k (because they are induction variables), while the mode κ validates the leaf node X (because it is a loop invariant). Therefore, our example has three kernel

[2] Keep in mind that $\kappa\sigma$ denotes a tree whose leaf is κ.

subtrees. Now, observe that each kernel subtree occurs on some path of loop nesting. For our example, each occurs in the inner loop, which corresponds to the path $(kloop, jloop, iloop)$, if we list the loops from innermost to outermost. Observe that, with respect to each loop along this path, a kernel subtree in some reference mode M either corresponds to an instance of κ (when the kernel subtree is invariant of that loop) or to an instance of M. This means that, to each kernel subtree, we can write a string of modes. For example, for the kernel subtree i we write $\kappa\kappa\sigma$; for k we write $\sigma\kappa\kappa$; and for X we write $\kappa\kappa\kappa$. Then, instantiate each of the modes appropriately (see [24] for more detail). In our example, we will replace the kernel subtree i, with the mode tree $\kappa_N\kappa_M\sigma_{1,L}$. Figure 4(c) shows the result of updating kernel subtrees. Observe, however, that the resulting tree is not yet a mode tree, because it has expression operations as internal nodes.

The final step applies a series of simplification rules to remove expression operations. For example, the addition a κ to any other tree t behaves identically to t alone; the κ does not alter the sequence of memory locations touched by t. Thus we can correctly replace $(+\ \kappa\ t)$ with t. Multiplying by a κ changes the reference pattern by expanding its height (if before a tree references 100 memory locations, now it will reference $100k$). Applying the latter rule element-wise to the $*$ subtree in Figure 4(c) yields Figure 4(d). Applying the $+$ rule once again finally yields a mode tree, given in Figure 4(e).

4 Performance Semantics

from Modes to Performance

Once we have a mode tree, the next step is to determine how this mode tree performs under various circumstances. For example, what are the implications of using an implementation $\sigma_{10^3,10^3}\kappa_{100}$ versus $\sigma_{10^3,10}\kappa_{100}\sigma_{10^4,100}$?[3] Our system predicts the performance of mode trees by composing the models for the constituents of a tree. It generates the constituent models from data it collects in a set of experiments. We call this experimentation process *mode scoping*.

Mode scoping determines how the performance of a mode instance m in a mode tree T varies. The performance of m depends not only on its own parameters (such as the swath-width of ρ, or the step of σ), but also on its *context* in T. The remainder of this section describes how:

1. we augment each mode instance to account for contextual effects
2. the system runs experiments to sweep this augmented parameter space
3. the system generates *kernel formulas* which predict the performance of modes-in-context
4. the system predicts the performance of a mode tree by instantiating and composing kernel formulas.

Our system, driven by specifications from (1), autonomously performs the operations (2) and (3), *once per architecture*. Then, once per candidate implementation, it performs operation (4).

[3] Observe that the latter corresponds to the *blocked* version of the former, with a block/tile size of 10. Section 5 discusses transformations.

4.1 Contextual Effects

We consider two aspects of the *context* of m in T: the performance of m may depend on, firstly, its position in T, and, secondly, on certain attributes of its neighbors. For example, σ's first parameter, the step distance, is sometimes an important determinant of performance, but other times not. This distinction, whether step is important, depends on context. To elaborate, we compare three mode trees. The first tree is the singleton $\sigma_{10^2,10^2}$. The second and third trees compose two mode instances: $\sigma_{10^2,10^2}\kappa_{10^2}$ and $\sigma_{10^2,10^2}\sigma_{10^4,10^2}$. Observe that each of these three mode trees is the result of composing a common instance, $\sigma_{10^2,10^2}$, with a variety of other instances. On a 400MHz Mobile Pentium II, the three trees take 7, 3.6, and 54 cycles per iteration, respectively.

To account for the effect of context, we *augment* a mode instance by summaries of its neighbors' attributes and by its position. We accomplish the former via *isolation attributes* and the latter via *position modes*. In summary, to account for contextual effects, we define the notion of a *mode-in-context*. The mode-in-context analog to a mode M in position P, $C_{P,M}$, is:

$$C_{P,M} = \langle P, M, I \rangle$$

where I is the subset of M's isolation attributes pertinent for P (e.g. child isolation attributes are not pertinent to leaf nodes). We now expand on these two augmentations.

Isolation Attributes. To account for the effect of parent-, child-, and sibling-parameters on performance, we augment each mode's parameter set by a set of *isolation attributes*. An isolation attribute encapsulates the following observation: the role a neighbor plays often does not depend on its precise details. Instead, the neighbor effect typically depends on coarse summaries of the surrounding nodes in the tree. For example, we have found that ρ is oblivious to whether it's child is κ_{10^6} versus $\sigma_{1,10^6}$. Yet ρ is sensitive to the width of its children (10^6 in both cases). Hence, we assign ρ an isolation attribute of `(child . width)`.[4] Table 1 shows the isolation attributes that we currently define.

Position Modes. We distinguish four position modes based on the following observation. For a mode instance m in mode tree T, the effect of m's parameters and its isolation attributes on performance often varies greatly depending on m's *position* in T. We thus define four position modes: *leaf*, *root*, *inner*, and *singleton*. These three correspond to the obvious positions in a tree.

[4] Notice that by stating "ρ isolates child-width", we enable *compositional* model generation with a bounded set of experiments. Isolation parameters essentially anonymize the effect of context (i.e. child being ρ doesn't matter); they permit the system to run experiments on these summary parameters, instead of once for every possible combination of child subtrees and parent paths.

Table 1. Isolation attributes for the three locality modes for parents, children, and siblings. Currently, we do not model sibling interactions.

mode	parent	child	siblings
κ	width	width	—
σ	reuse, width	width	—
ρ	—	width	—

4.2 Mode Scoping

To determine the performance of each mode-in-context, the system runs experiments. We call this experimentation *mode scoping*; it is akin to the well-known problem of parameter sweeping. The goal of a parameter sweep is to discover the relationship of parameter values to the output value, the parameter curve. For a mode-in-context, $C_{P,M}$, mode scoping sweeps over the union of M's parameters and $C_{P,M}$'s isolation attributes.

Neither exhaustive nor random sweeping strategies suffices. It is infeasible to run a complete sweep of the parameter space, because of its extent. For example, the space for the κ mode contains 10^9 points; a complete sweep on a 600MHz Intel Pentium III would take 30 years. Yet, if performance is piecewise-linear, then the system need not probe every point. Instead, it looks for end points and inflection points. However, a typical planar cut through the parameter space has an inflection point population of around 0.1%. Thus, any random sampling will not prove fruitful.

Our sweeping strategy sweeps a number of planar slices through a mode-in-context's parameter space. The system uses a divide-and-conquer technique to sweep each plane, and a pruning-based approach to pick planes to sweep.[5] Our current implementation runs 10–20 experiments per plane (out of anywhere from thousands to millions of possible experiments). It chooses approximately 60 planes per dimension (one dimension per mode parameter or isolation attribute) in two passes. The first pass probes end points, uniform-random samples, and log_{20}- and log_2-random samples. The goal of the first pass is to discover possible inflection points. The second pass then runs experiments on combinations of discovered inflection points. Because the first pass may have run more experiments than discovered inflection points, the second pass first prunes the non-inflection points before choosing planes to sweep. The result of mode scoping is a mapping from parameter choices to actual performance for those choices.

4.3 Generating Kernel Formulas

After mode scoping a mode-in-context, the system then generates a model which predicts the observed behavior. We call these models *kernel formulas*, because

[5] A divide-and-conquer strategy will not discover associativity effects, because the effects of associativity do not vary monotonically with any one parameter. This is a subject of future research.

they are symbolic templates. Later, the system will instantiate these kernel formulas for the particulars of a mode tree. Instantiating a kernel formula is the straightforward process of variable substitution. For example, a kernel formula might be $3 + p_1^2 + i_2$ — meaning that this kernel formula is a function of the fist mode parameter and the second isolation parameter. To instantiate this kernel formula, simply substitute the actual values for p_1 and i_2.

Our system, similar to [4], uses linear regression to generate kernel formulas. To handle nonlinearities, the regression uses quadratic cross terms and reciprocals. Furthermore, it *quantizes* performance. That is, instead of generating models which predict the performance of 6 cycles per iteration versus 100 cycles per iteration, the system generates models which predict that performance is in, for example, the lowest or highest quantile. We currently quantize performance into five buckets.[6] The system now has one kernel formula per mode-in-context.

system	mode-in-context	kernel formula
Pentium	$\langle inner, \kappa \rangle$	$1 - 9.5e - 11p_0 + \frac{0.95}{p_0 i_0} + \frac{0.086}{\sqrt{p_0}}$
PA-RISC	$\langle inner, \kappa \rangle$	$2 + \frac{0.05}{p_0^2} + \frac{0.56}{p_0 i_0} + \frac{0.45}{i_0 i_1}$
Pentium	$\langle leaf, \sigma \rangle$	$2.6 + 5.6\frac{p_0}{10^4} - 6\frac{\sqrt{p_0 i_0}}{10^2} - 1.6\sqrt{p_0} - 5.8\frac{\sqrt{p_1}}{10^4 i_1} 5.2\frac{\sqrt{p_0 p_1}}{10^4}$
Pentium	$\langle root, \rho \rangle$	$0.98 - 8.4\frac{p_1}{10^7 i_0} - 8.5\frac{\sqrt{p_2}}{10^4 p_0} - \frac{0.76}{i_0^2} + 3\frac{\sqrt{p_1}}{10^3 i_0} + \frac{1.2}{\sqrt{i_0}}$
PA-RISC	$\langle root, \rho \rangle$	$2 - 7.5\frac{p_1}{10^7 i_0} + \frac{233}{p_2 i_0} - \frac{1.1}{i_0} + 0.001\frac{\sqrt{p_0}}{i_0} + 0.003\frac{\sqrt{p_1}}{i_0}$

Fig. 5. Some example kernel formulas for two machines: one with a 700MHz Mobile Pentium III, and the other with a 400MHz PA-RISC 8500.

4.4 Evaluating a Mode Tree

Finally, the system evaluates the performance of a mode tree by instantiating and composing kernel formulas. We describe these two tasks briefly. Recall that a mode tree T is a tree whose nodes are mode instances. First, then, for each mode instance $m \in T$, the system computes the mode-in-context for m's mode. This yields a collection of $C_{P,M}$. Next, the system computes the values for the isolation parameters of each $C_{P,M}$. Both of these steps can be accomplished with simple static analysis: position mode can be observed by an instance's location in the tree, and isolation attributes, as we have defined them in Sec. 4.1, can also be easily derived. The system now has all the information necessary to instantiate the kernel formulas for each $C_{P,M}$: substitute the actual values for mode parameters and isolation attributes into the kernel formula.

[6] Quantizing performance is critical to generating well-fitting models which generalize. For example, within a plane, we often observe step-wise relationships between performance and parameter values — this is typical behavior in systems with multiple levels of cache. With quantization, the system need not model this curve as a step function, which is very difficult with low-degree polynomials.

The second step in mode tree evaluation composes the instantiated kernel formulas. One could imagine a variety of composition operators, such as addition, multiplication, and maximum. In [24] we explored several of these operators experimentally. Not too surprisingly, we found that the best choices were maximum for parent-child composition, and addition for sibling composition.[7]

5 Transformations from Modes to (better) Modes

Program optimizations map a mode tree to a set of mode trees. The system is driven by any number of optimization specifications. Table 6 provides a specification for tiling. Each rule gives a template of the mode trees to which the transformation applies (*domain*), and the resultant mode tree (*range*). Notice that the transformation specification gives names to the important mode parameters and optimization parameters (like tile size). The domain template trees use capital letters to name context. In [24], we describe in detail how the a specification describes context, and how our system determines the set of all possible applications of a transformation to a mode tree (e.g. tiling may apply in many ways to a single mode tree; perhaps as many as once per loop in the nest).

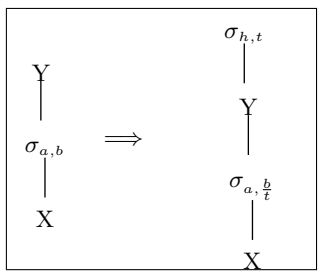


Fig. 6. The transformation specification for tiling: $domain \Longrightarrow range$.

For example, consider loop tiling. Tiling is possibly beneficial whenever a mode tree contains at least one σ, but contains no ρ — as commonly formulated, tiling requires that all references and loop bounds be affine functions of the enclosing loops' induction variables. To any mode tree which satisfies tiling's domain criteria, corresponds the set of mode trees which result from tiling one of the loops in the original implementation.

6 Experiments

We now provide some initial performance numbers. We compare the predicted and actual performance of several implementations. The predicted numbers come

[7] Recall that our kernel formulas represent cycles per iteration, rather than cycles. Hence, maximum is a good choice for parent-child composition. Had kernel formulas represented cycles, then multiplication would likely have been the best choice.

from our system, using the performance evaluation strategy described in Sec. 4; the input to the system is the mode tree corresponding to an implementation choice. The actual numbers come from running that actual implementation on a real machine — in this case a 700MHz Mobile Pentium III. In this initial study, we look at loop interchange.

6.1 Loop Interchange

codes	modes

do $i = 1, M$
  do $j = 1, N, s$
  $\ldots A(B(i) + j) \ldots$
	$T_1 = \sigma_{s,N}\rho_{?_1,?_2,M}$

do $j = 1, N, s$
  do $i = 1, M$
  $\ldots A(B(i) + j) \ldots$
	$T_2 = \rho_{?_1,?_2,M}\sigma_{s,N}$

(a)

		T_1/T_2	
S	$?_1$,$?_2$	pred.	act.
1	10^3	0.84	0.38
1	10^6	0.26	0.64
10	10^6	0.56	0.78
10^3	10^3	1.56	5.68
10^3	10^6	0.94	1
10^3	10^2	1.63	6.12

(b)

Fig. 7. Comparing the two versions of a loop nest. The ratio of T_1 to T_2 is the ratio of predicted performance for that row's parameter value assignment. Notice that both cases have an equal number of memory references.

Figure 7 shows two loop nests, identical in structure except that the loops have been interchanged. Which of the two versions performs better? Phrased in our mode language, we would ask this question: under what conditions will $\sigma\rho$ outperform $\rho\sigma$? The table in Fig. 8(b) shows this comparison for several choices of mode parameters, for both actual runs and for performance as predicted by our system. This table shows T_1/T_2, which stands for the performance of the implementation represented by mode tree T_1 versus that of T_2. Thus if $T_1/T_2 < 1$ choose implementation T_1, if $T_1/T_2 > 1$, choose T_2, and if $T_1/T_2 = 1$ then the choice is a wash. Observe that for the cases shown in the table, the prediction would always make the correct choice. Figure 8 and shows a similar study with one σ instance versus another.

7 Related Work

Finally, we present research which has inspired our solution. We summarize these works into the following four groups:

codes	modes

do $i = 1, M$
  do $j = 1, N$
  $\ldots A(i * R + j * S) \ldots$
	$T_1 = \sigma_{S,N}\sigma_{R,M}$

do $j = 1, N$
  do $i = 1, M$
  $\ldots A(i * R + j * S) \ldots$
	$T_2 = \sigma_{R,M}\sigma_{S,N}$

(a)

		T_1/T_2	
S	R	pred.	act.
1	1	1	1
1	2	0.67	1
1	5	0.52	0.75
1	10	0.46	0.48
1	100	0.43	0.22
1	1000	0.61	0.61

(b)

Fig. 8. Another comparison of two implementations of a simple loop. The ratio of T_1 to T_2 is the ratio of predicted performance for that row's parameter value assignment; therefore a lower ratio means that the second implementation, T_2, is a better choice than the first. For every choice of S and R, we chose $N = M = 10^3$.

Combined static-dynamic approaches: Given user-specified performance templates, Brewer [4] derives platform-specific cost models (based on profiling) to guide program variant choice. The FFTW project optimizes FFTs with a combination of static modeling (via dynamic programming) and experimentation to choose the FFT algorithm best suited for an architecture [12]. Gatlin and Carter introduces *architecture cognizance*, a technique which accounts for hard-to-model aspects of the architecture [13]. Lubeck *et al.* [21] use experiments to develop a hierarchical model which predict the contribution of each level of the memory hierarchy to performance.

Adaptive optimizations: Both Saavedra and Park [26] and Diniz and Rinard [7] adapt programs to knowledge discovered while the program is running. Voss and Eigenmann describe ADAPT [32], a system which can dynamically generate and select program variants. A related research area is dynamic compilation and program specialization, from its most abstract beginnings by Ershov [9], to more recent work, such as [8,6,20,15].

System scoping: McCalpin's STREAM benchmark discovers the *machine balance* of an architecture via experimentation [22]. In addition to bandwidth, McVoy's and Staelin's lmbench determines a set of system characteristics, such as process creation costs, and context switching overhead [23]. Saavedra and Smith use *microbenchmarks* to experimentally determine aspects of the system [27]. Gustafson and Snell [16] develop a scalable benchmark, HINT, that can accurately predict a machine's performance via its memory reference capacity.

Automation: Collberg automatically generates a compiler back-end by discovering many aspects of the underlying system via experimentation [5]. Hoover and Zadeck use architectural specifications to automatically generate a compiler back-end tuned for that architecture [17]. The Sharlit toolkit automatically generates dataflow optimizers based on specifications [31].

8 Conclusion

In an ideal world, static analysis would not only suffice, but would not limit the universe of approachable input codes. Unfortunately, we have experienced situations which break with this ideal, on one or both fronts: either static analysis fails to provide enough information to make good transformation decisions, or the static analyses themselves preclude the very codes we desire to optimize. This paper has presented a modeling methodology which tackles these two problems.

References

1. D. Bailey and et al. NAS parallel benchmarks. http://science.nas.nasa.gov/Software/NPB.
2. D. A. Berson, R. Gupta, and M. L. Soffa. URSA: A unified resource allocator for registers and functional units in vliw architectures. In *Conference on Architectures and Compilation Techniques for Fine and Medium Grain Parallelism*, Orlando, FL, Jan. 1993.
3. J. Bilmes, K. Asanović, C.-W. Chin, and J. Demmel. Optimizing matrix multiply using PHiPAC: a portable, high-performance, ANSI C coding methodology. In *International Conference on Supercomputing*, 1997.
4. E. A. Brewer. *Portable High-Performance Supercomputing: High-Level Platform-Dependent Optimization*. PhD thesis, Massachusetts Institute of Technology, 1994.
5. C. S. Collberg. Reverse interpretation + mutation analysis = automatic retargeting. In *Programming Language Design and Implementation*, 1997.
6. C. Consel, L. Hornof, J. Lawall, R. Marlet, G. Muller, J. Noyé, S. Thibault, and N. Volanschi. Tempo: Specializing systems applications and beyond. In *Symposium on Partial Evaluation*, 1998.
7. P. Diniz and M. Rinard. Dynamic feedback: An effective technique for adaptive computing. In *Programming Language Design and Implementation*, June 1997.
8. D. R. Engler, W. C. Hsieh, and M. F. Kaashoek. 'C: A language for high-level, efficient, and machine-independent dynamic code generation. In *Principles of Programming Languages*, Saint Petersburg, FL, Jan. 1996.
9. A. P. Ershov. On the partial computation principle. *Inf. Process. Lett.*, 1977.
10. J. Ferrante, V. Sarkar, and W. Thrash. On estimating and enhancing cache effectiveness. In *Workshop on Languages and Compilers for Parallel Computing*, 1991.
11. B. Fraguela, R. Doallo, and E. Zapata. Automatic analytical modeling for the estimation of cache misses. In *Parallel Architectures and Compilation Techniques*, Oct. 1999.
12. M. Frigo and S. G. Johnson. The fastest fourier transform in the west. Technical Report MIT-LCS-TR-728, Massachusetts Institute of Technology, Laboratory for Computer Science, Sept. 1997.
13. K. S. Gatlin and L. Carter. Architecture-cognizant divide and conquer algorithms. In *Supercomputing*, Nov. 1999.
14. S. Ghosh. *Cache Miss Equations: Compiler Analysis Framework for Tuning Memory Behavior*. PhD thesis, Princeton, Sept. 1999.

15. B. Grant, M. Mock, M. Philipose, C. Chambers, and S. J. Eggers. DyC: An expressive annotation-directed dynamic compiler for c. Technical Report UW-CSE-97-03-03, University of Washington, Department of Computer Science and Engineering, June 1998.
16. J. L. Gustafson and Q. O. Snell. HINT–a new way to measure computer performance. In *HICSS-28*, Wailela, Maui, Hawaii, Jan. 1995.
17. R. Hoover and K. Zadeck. Generating machine specific optimizing compilers. In *Principles of Programming Languages*, St. Petersburg, FL, 1996.
18. W. Kelly and W. Pugh. A unifying framework for iteration reordering transformations. In *Proceedings of IEEE First International Conference on Algorithms and Architectures for Parallel Processing*, Apr. 1995.
19. R. E. Ladner, J. D. Fix, and A. LaMarca. Cache performance analaysis of traversals and random accesses. In *Symposium on Discrete Algorithms*, Jan. 1999.
20. M. Leone and P. Lee. A declarative approach to run-time code generation. In *Workshop on Compiler Support for System Software*, pages 8–17, Tuscon, AZ, 1996.
21. O. M. Lubeck, Y. Luo, H. J. Wasserman, and F. Bassetti. Development and validation of a hierarhical memory model incorporating cpu- and memory-operation overlap. Technical Report LA-UR-97-3462, Los Alamos National Laboratory, 1998.
22. J. D. McCalpin. Memory bandwidth and machine balance in current high performance computers. In *IEEE Computer Society Technical Committee on Computer Architecture Newsletter*, Dec. 1995.
23. L. McVoy and C. Staelin. lmbench: Portable tools for performance analysis. In *Usenix Proceedings*, Jan. 1995.
24. N. Mitchell. *Guiding Program Transformations with Modal Performance Models.* PhD thesis, University of California, San Diego, Aug. 2000.
25. N. Mitchell, K. Högstedt, L. Carter, and J. Ferrante. Quantifying the multi-level nature of tiling interactions. In *International Journal on Parallel Programming*, June 1998.
26. R. H. Saavedra and D. Park. Improving the effectiveness of software prefetching with adaptive execution. In *Parallel Architectures and Compilation Techniques*, Boston, MA, Oct. 1996.
27. R. H. Saavedra and A. J. Smith. Measuring cache and TLB performance and their effect on benchmark run times. *IEEE Trans. Comput.*, 44(10):1223–1235, Oct. 1995.
28. V. Sarkar. Automatic selection of high-order transformations in the IBM XL FORTRAN compilers. *IBM J. Res. Dev.*, 41(3), May 1997.
29. V. Sarkar and R. Thekkath. A general framework for iteration-reordering loop transformations (Technical Summary). In *Programming Language Design and Implementation*, 1992.
30. O. Temam, E. D. Granston, and W. Jalby. To copy or not to copy: A compile-time technique for assessing when data copying should be used to eliminate cache conflicts. In *Supercomputing '93*, pages 410–419, Portland, Oregon, Nov. 1993.
31. S. W. K. Tjiang and J. L. Hennessy. Sharlit—a tool for building optimizers. In *Programming Language Design and Implementation*, pages 82–93, San Francisco, California, June 1992. *SIGPLAN Notices* 27(7), July 1992.
32. M. J. Voss and R. Eigenmann. ADAPT: Automated de-coupled adaptive program transformation. In *International Conference on Parallel Processing*, Toronto, CA, Aug. 2000.

33. T. P. Way and L. L. Pollock. Towards identifying and monitoring optimization impacts. In *Mid-Atlantic Student Workshop on Programming Languages and Systems*, 1997.
34. R. C. Whaley and J. Dongarra. Automatically tuned linear algebra software. In *Supercomputing*, Nov. 1998.
35. D. Whitfield and M. L. Soffa. An approach to ordering optimizing transformations. In *Principles and Practice of Parallel Programming*, pages 137–146, Seattle, WA, Mar. 1990.

Fast Automatic Generation of DSP Algorithms

Markus Püschel[1], Bryan Singer[2], Manuela Veloso[2], and José M.F. Moura[1]

[1] Carnegie Mellon University, Pittsburgh
Department of Electrical and Computer Engineering
{moura,pueschel}@ece.cmu.edu
[2] Department of Computer Science
{bsinger,veloso}@cs.cmu.edu

Abstract. SPIRAL is a generator of optimized, platform-adapted libraries for digital signal processing algorithms. SPIRAL's strategy translates the implementation task into a search in an expanded space of alternatives. These result from the many degrees of freedom in the DSP algorithm itself and in the various coding choices. This paper describes the framework to represent and generate efficiently these alternatives: the formula generator module in SPIRAL. We also address the search module that works in tandem with the formula generator in a feedback loop to find optimal implementations. These modules are implemented using the computer algebra system GAP/AREP.

1 Introduction

SPIRAL, [1], is a system that generates libraries for digital signal processing (DSP) algorithms. The libraries are generated at installation time and they are optimized with respect to the given computing platform. When the system is upgraded or replaced, SPIRAL can regenerate and thus readapt the implementations. SPIRAL currently focuses on DSP transforms including the discrete trigonometric transforms, the discrete Fourier transform, and several others. Other approaches to similar problems include for DSP transforms [2] and for other linear algebra algorithms [3,4,5,6].

SPIRAL generates a platform-adapted implementation by searching in a large space of alternatives. This space combines the many degrees of freedom associated with the transform and the coding options.

The architecture of SPIRAL is displayed in Figure 1. The DSP transform specified by the user is input to a formula generator block that generates one out of many possible formulas. These formulas are all in a sense equivalent: barring numerical errors, they all compute the given transform. In addition, they all have basically the same number of floating point operations. What distinguishes them is the data flow pattern during computation, which causes a wide range of actual runtimes. The output of the formula generator is a formula given as a program in a SPIRAL proprietary language called SPL (signal processing language). The SPL program is input to the SPIRAL-specific formula translator block that compiles it into a C or Fortran program [7]. This program, in turn, is compiled

V.N. Alexandrov et al. (Eds.): ICCS 2001, LNCS 2073, pp. 97–106, 2001.
© Springer-Verlag Berlin Heidelberg 2001

by a standard C or Fortran compiler. The runtime of the resulting code is then fed back through a search module. The search module controls the generation of the next formulas to be tested using search and learning techniques (Section 4). Iteration of this loop yields a platform-adapted implementation.

This paper focuses on the formula generator and its interplay with the search module. It explains the underlying mathematical framework (Section 2) and its implementation (Section 3) using the computer algebra system GAP/AREP [8].

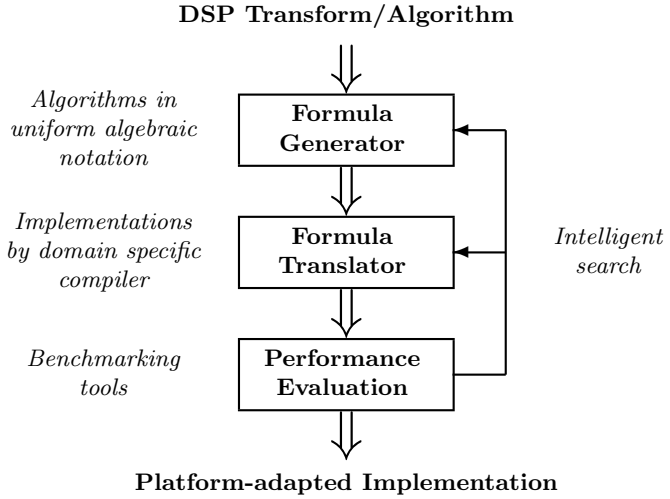


Fig. 1. The architecture of SPIRAL.

2 DSP Transforms and Algorithms

In this section we introduce the framework used by SPIRAL to describe linear DSP (digital signal processing) transforms and their fast algorithms. A similar approach has been used in [9] for the special case of FFT algorithms. We start with an introductory example.

2.1 Example: DFT, Size 4

The DFT (discrete Fourier transform) of size 4 is given by the following matrix DFT_4, which is then factored as a product of sparse structured matrices.

$$\mathrm{DFT}_4 = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & i & -1 & -i \\ 1 & -1 & 1 & -1 \\ 1 & -i & -1 & i \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & -1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & i \end{bmatrix} \begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & -1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & -1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

This factorization is an example of a fast algorithm for DFT_4. Using the Kronecker (or tensor) product of matrices, $\otimes$, and introducing symbols L^4_2 for the permutation matrix (right-most matrix), and $\mathrm{T}^4_2 = \mathrm{diag}(1, 1, 1, i)$, the algorithm can be written in the very concise form

$$\mathrm{DFT}_4 = (\mathrm{DFT}_2 \otimes \mathrm{I}_2) \cdot \mathrm{T}^4_2 \cdot (\mathrm{I}_2 \otimes \mathrm{DFT}_2) \cdot \mathrm{L}^4_2 . \tag{1}$$

The last expression is an instantiation of the celebrated Cooley-Tukey algorithm [10], also referred to as the fast Fourier transform (FFT).

2.2 Transforms, Rules, Ruletrees, and Formulas

Transforms: A (linear) DSP transform is a multiplication of a vector x (the sampled signal) by a certain $(n \times n)$-matrix M (the transform), $x \mapsto M \cdot x$. The transform is denoted by a symbol having the size n of the transform as a subscript. Fixing the parameter "size" determines a special instance of the transform, e.g., DFT_4 denotes a DFT of size 4. For arbitrary size n, the DFT is defined by

$$\mathrm{DFT}_n = [w_n^{k\ell} \mid k, \ell = 0..n-1],$$

where $w_n = e^{2\pi j/n}$ denotes an nth root of unity. In general, a transform can have other determining parameters rather than just the size. Transforms of interest include the discrete cosine transforms (DCT) of type II and type IV,

$$\mathrm{DCT}_n^{(\mathrm{II})} = [\cos((\ell + 1/2)k\pi/n) \mid k, \ell = 0 \ldots n-1], \text{ and}$$
$$\mathrm{DCT}_n^{(\mathrm{IV})} = [\cos((k + 1/2)(\ell + 1/2)\pi/n) \mid k, \ell = 0 \ldots n-1],$$

which are used in the current JPEG and MPEG standards, [11], respectively (given here in an unscaled version). Further examples with different areas of application are the other types of discrete cosine and sine transforms (DCTs and DSTs, type I – IV), the Walsh-Hadamard transform (WHT), the discrete Hartley transform (DHT), the Haar transform, the Zak transform, the Gabor transform, and the discrete wavelet transforms.

Breakdown Rules: All of the transforms mentioned above can be evaluated using $O(n \log n)$ arithmetic operations (compared to $O(n^2)$ operations required by a straightforward implementation). These algorithms are based on sparse structured factorizations of the transform matrix. For example, the Cooley-Tukey FFT is based on the factorization

$$\mathrm{DFT}_n = (\mathrm{DFT}_r \otimes \mathrm{I}_s) \cdot \mathrm{T}_s^n \cdot (\mathrm{I}_r \otimes \mathrm{DFT}_s) \cdot \mathrm{L}_r^n, \tag{2}$$

where $n = r \cdot s$, L_r^n is the stride permutation matrix, and T_s^n is the twiddle matrix, which is diagonal (see [12] for details). We call an equation like (2) a *breakdown rule*, or simply *rule*. A breakdown rule

- is an equation that factors a transform into a product of sparse structured matrices;
- may contain (possibly different) transforms of (usually) smaller size;
- the applicability of the rule depends on the parameters (e.g., size) of the transform.

Examples of breakdown rules for $\mathrm{DCT}^{(\mathrm{II})}$ and $\mathrm{DCT}^{(\mathrm{IV})}$ are

$$\mathrm{DCT}_n^{(\mathrm{II})} = P_n \cdot (\mathrm{DCT}_{n/2}^{(\mathrm{II})} \oplus \mathrm{DCT}_{n/2}^{(\mathrm{IV})}) \cdot P'_n \cdot (\mathrm{I}_{n/2} \otimes \mathrm{DFT}_2) \cdot P''_n, \text{ and}$$
$$\mathrm{DCT}_n^{(\mathrm{IV})} = S_n \cdot \mathrm{DCT}_n^{(\mathrm{II})} \cdot D_n,$$

where P_n, P'_n, P''_n are permutation matrices, S_n is bidiagonal, and D_n is a diagonal matrix (see [13] for details).

A transform usually has several different rules. Rules for the DFT that we can capture from fast algorithms as they are given in literature, include the Cooley-Tukey rule ($n = r \cdot s$ composite), Rader's rule (n prime), Good-Thomas rule ($n = r \cdot s$, $\gcd(r, s) = 1$), and several others (see [12]).

Besides breakdown rules, SPIRAL includes also rules for base cases, such as

$$\mathrm{DFT}_2 = \left[\begin{smallmatrix} 1 & 1 \\ 1 & -1 \end{smallmatrix}\right],$$

which shows that a DFT_2 can be computed with 2 additions/subtractions.

Formulas and Ruletrees: A formula is a mathematical expression that represents a sparse structured factorization of a matrix of fixed size. A formula is composed of mathematical operators (like $\cdot, \oplus, \otimes$), basic constructs (permutation, diagonal, plain matrix), symbolically represented matrices (like I_5 for an identity matrix of size 5), and transforms with fixed parameters. An example is

$$(\mathrm{DFT}_4 \otimes \mathrm{diag}(1, 7)) \cdot \left(\mathrm{I}_6 \oplus \left[\begin{smallmatrix} 1 & 4 \\ 2 & -1 \end{smallmatrix}\right]\right). \tag{3}$$

We call a formula *fully expanded* if it does not contain any transforms.

Expanding a transform M of a given size using one of the applicable rules creates a formula, which (possibly) contains transforms of smaller size. These, in turn, can be expanded further using the same or different rules. After all transforms have been expanded, we obtain a formula that represents in a unique way a fast algorithm for M. Since the formula is uniquely determined by the rules applied in the different stages, we can represent a formula, and hence an algorithm, by a tree in which each node is labeled with a transform (of size) and the rule applied to it. A node has as many children as the rule contains smaller transforms (e.g., the Cooley-Tukey rule (2) gives rise to binary trees). We call such a tree a *ruletree*. The ruletree is fully expanded, if all its leaves are base cases. Thus, within our framework,

fully expanded ruletree = fully expanded formula = algorithm.

An example for a fully expanded ruletree for a $\mathrm{DCT}_8^{(\mathrm{IV})}$ is given in Figure 2 (we omitted the rules for the base cases). The rules' identifiers used are not of significance.

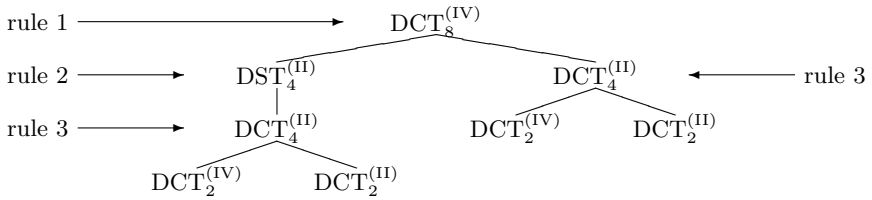


Fig. 2. A ruletree for $\mathrm{DCT}^{(\mathrm{IV})}$, size 8

2.3 The Formula Space

Applying different rules in different ways when expanding a transform gives rise to a surprisingly large number of mathematically equivalent formulas. Applying only the Cooley-Tukey rule (2) to a DFT of size $n = 2^k$ gives rise to $\Theta(5^k/k^{3/2})$ many different formulas. This large number arises from the degree of freedom in splitting 2^k into 2 factors. Using different rules and combinations thereof leads to exponential growth (in n) in the number of formulas. As an example, the current implementation of the formula generator contains 13 transforms and 31 rules and would produce about 10^{153} different formulas for the $\mathrm{DCT}^{(\mathrm{IV})}_{512}$.

By using only the best rules available (regarding the number of additions and multiplications), the algorithms that can be derived all have about the same arithmetic cost. They differ, however, in their data access during computation, which leads to very different runtime performances. As an example, Figure 3 shows a histogram of runtimes for all 31,242 formulas generated with our current set of rules for a $\mathrm{DCT}^{(\mathrm{IV})}_{16}$. The histogram demonstrates that even for a transform of small size, there is a significant spread of running times, more than a factor of two from the fastest to the slowest. Further, it shows that there are relatively few formulas that are amongst the fastest.

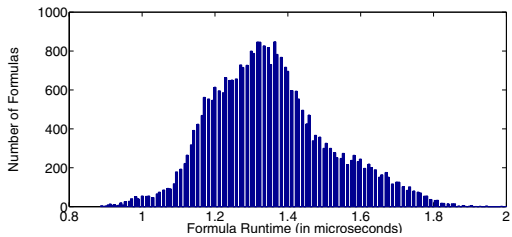


Fig. 3. Histogram of running times for all 31,242 $\mathrm{DCT}^{(\mathrm{IV})}$, size 2^4, formulas generated by SPIRAL's formula generator on a Pentium III running Linux.

3 The Formula Generator

Briefly, the formula generator is a module that produces DSP algorithms given as formulas for a user specified transform of given size. The formula generator is coupled with a search module that uses a feedback loop of formula generation and evaluation to optimize formulas with respect to a given performance measure.

Formula generation and formula manipulation fall into the realm of symbolic computation, which lead us to choose the language and computer algebra system GAP [8], including AREP [14], as an implementation platform. GAP provides the infrastructure for symbolic computation with a variety of algebraic objects. The GAP share package AREP is particularly focused on structured matrices and their symbolic manipulation. A high level language like GAP with its readily available functionality facilitates the implementation of our formula generator. It provides, as an additional advantage, *exact* arithmetic for square roots, roots of unity, and trigonometric expressions that make up the entries of most DSP

transforms and formulas. The current implementation of the formula generator has about 12,000 lines of code.

The main objectives for the implementation of the formula generator are

- **efficiency:** it should generate formulas *fast* and store them efficiently; this is imperative since the optimization process requires the generation of many formulas;
- **extensibility:** it should be easy to expand the formula generator by including new transforms and new rules.

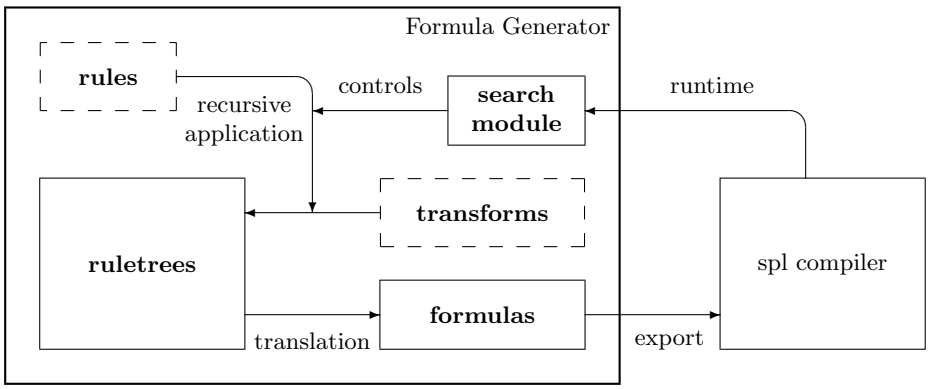


Fig. 4. Internal architecture of the formula generator including the search module. The main components are recursive data types for representing ruletrees and formulas, and extensible data bases (dashed boxes) for rules and transforms.

The architecture of the formula generator and the process of formula generation is depicted in Figure 4. We start with a transform with given parameters as desired by the user, e.g., a $\mathrm{DCT}_{64}^{(\mathrm{II})}$. The transform is recursively expanded into a ruletree. The choice of rules is controlled by the search module (see Section 4). The ruletree then is converted into a formula, which, in turn, is exported as an SPL program. The SPL program is compiled into a Fortran or C program (see [7]). The runtime of the program is returned to the formula generator. Based on the outcome, the search module triggers the derivation of different ruletrees.

By replacing the spl compiler block in Figure 4 by another evaluation function, the formula generator becomes a potential optimization tool for DSP algorithms with respect to other performance measures. Examples of potential interest include numerical stability or critical path length.

As depicted in Figure 4, and consistent with the framework presented in Section 2, the main components of the formula generator are formulas, transforms, rules, and ruletrees. Formulas and ruletrees are objects meant for computation and manipulation, and are realized as recursive data types. Transforms and rules are merely collections of information needed by the formula generator. We elaborate on this in the following. The search module is explained in Section 4.

Formulas: Formulas are implemented by the recursive data type SPL. We chose the name SPL since it is similar to the language SPL understood by

the formula translator (see Section 1). A formula is an instantiation of SPL and is called spl. An spl is a GAP record with certain fields mandatory to all spls. Important examples are the fields `dimensions`, which gives the size of the represented matrix, and the field `type`, which contains a string indicating the type of spl, i.e, node in the syntax tree. Basic types are `diag` for diagonal matrices or `perm` for permutation matrices. Examples for composed types are `tensor` or `directSum`. The type `symbol` is used to symbolically represent frequently occurring matrices such as identity matrices I_n. The list of symbols known to the formula generator can be extended. A complete overview of all types is given in Table 1.

Table 1. The data type SPL in Backus-Naur form as the disjoint union of the different types. The string identifying the type is given in double quotes

```
<spl> ::= <matrix>                                  ; "mat"
        | <diagonal>                                ; "diag"
        | <permutation>                             ; "perm"
        | Symbol( <name>, <params> )                ; "symbol"
        | NonTerminal( <name>, <params> )           ; "nonTerminal"
        | <spl> * .. * <spl>                        ; "compose"
        | DirectSum(<spl>, .., <spl>)               ; "directSum"
        | TensorProduct(<spl>, .., <spl>)           ; "tensor"
        | <scalar> * <spl>                          ; "scalarMultiple"
        | <spl> ^ <"perm"-spl>                      ; "conjugate"
        | <spl> ^ <positive-int>                    ; "power"
```

The data type SPL mirrors the language SPL (Section 1) with the exception of the type `nonTerminal`. A `nonTerminal` spl represents a transform of fixed size, e.g., DFT_{16}, within a formula. The non-terminal spls available depend on the global list of transforms, which is explained below.

Other fields are specific to certain types. For example, an spl of type `diag` has a field `element` that contains the list of the diagonal entries; an spl of type `compose` has a field `factors` containing a list of spls, which are the factors in the represented product. For each of the types a function is provided to construct the respective spls. As an example, we give the spl corresponding to the formula in (3) as it is constructed in the formula generator.

```
ComposeSPL(TensorSPL(SPLNonTerminal("DFT", 4), SPLDiag([1, 7])),
           DirectSumSPL(SPLSymbol("I", 6), SPLMat([[1, 4], [2, -1]])))
```

Transforms: All transforms known to the formula generator are contained in the global list `NonTerminalTable`. Each entry of the list is a record corresponding to one transform (e.g., DFT). The record for a transform M stores the necessary information about M. Important fields include (1) `symbol`, a string identifying M (e.g, "DFT"); (2) `CheckParams`, a function for checking the validity of the parameters used to create an instantiation of M, usually the parameter is just the size, but we allow for arbitrary parameters; (3) `TerminateSPL`, a function to convert an instantiation of M into a plain matrix (type `mat`), used for verification. An instantiation of a transform (e.g., a DFT_{16}) is created as an spl of type

`nonTerminal` as explained in the previous paragraph. The transform table can easily be extended by supplying this record for the new transform to be included.

Rules: All breakdown rules known to the formula generator are contained in the global list `RuleTable`. Each entry of the list corresponds to one rule (e.g., Cooley-Tukey rule). Similar to the transforms, rules are records storing all necessary information about the rule. Important fields of a rule R include (1) `nonTerminal`, the symbol of the transform R applies to (e.g., "DFT"); (2) `isApplicable`, a function checking whether R is applicable to a transform with the given parameters (e.g., Cooley-Tukey is applicable iff n is not prime); (3) `allChildren`, a function returning the list of all possible children configurations for R given the transform parameters, children are non-terminal spls; (4) `rule`, the actual rule, given the parameters for transform, returns an spl.

The rule table can also easily be extended by supplying this record for the new rule to be included.

Ruletrees: A ruletree is a recursive data type implemented as a record. Important fields include (1) `node`, the non-terminal spl expanded at the node; (2) `rule`, the rule used for expansion at the node; (3) `children`, ordered list of children, which again are ruletrees. In addition, we allow for a field `SPLOptions` that controls implementation choices that cannot be captured on the formula, i.e., algorithmic, level. An example is the code unrolling strategy. Setting `SPLOptions` to `"unrolling"` causes the code produced from the entire subtree to be unrolled.

There are two main reasons for having ruletrees as an additional data structure to formulas (both represent DSP algorithms): (1) ruletrees require much less storage than the corresponding formulas (a ruletree only consists of pointers to rules and transforms) and can be generated very fast, thus moving the bottleneck in the feedback loop (Figure 4) to the spl compiler; and (2) the search algorithms (see Section 4) use the ruletree data structure to easily derive variations of algorithms in the optimization process.

Infrastructure: In addition to these data types, the formula generator provides functionality for their manipulation and investigation. Examples include functions that (1) convert ruletrees into formulas; (2) export formulas as SPL programs; (3) convert formulas into plain matrices; (4) verify rules (for given transforms) and formulas using exact arithmetic where possible; (5) compute an upper bound for the arithmetic cost of an algorithm given as a formula.

4 Search

In this section, we discuss the search module shown in Figure 4 and how it interfaces with the formula generator. Given that there is a large number of formulas for any given signal transform, an important problem is finding a formula that runs as fast as possible. Further, the runtimes of formulas for a given transform vary widely as shown in Figure 3. Unfortunately, the large number of formulas for any given signal transform makes it infeasible to exhaustively time every formula for transforms of even modest sizes. Thus, it is crucial to *intelligently* search the space of formulas. We have implemented the following search methods.

Exhaustive Search: Determines the fastest formula, but becomes infeasible even at modest transform sizes since there is a large number of formulas.

Dynamic Programming: A common approach has been to use dynamic programming (DP) [15]. DP maintains a list of the fastest formulas it has found for each transform and size. For a particular transform and its applicable rules, DP considers all possible sets of children. For each child, DP substitutes the best ruletree found for that transform. DP makes the assumption that the fastest ruletree for a particular transform is also the best way to split a node of that transform in a larger tree. For many transforms, DP times very few formulas and still is able to find reasonably fast formulas.

Random Search: A very different approach is to generate a fixed number of random formulas and time each. This approach assumes that there is a sufficiently large number of formulas that have runtimes close to the optimal.

STEER: As a refinement to random search, we have developed an evolutionary stochastic search algorithm called STEER [16]. STEER is similar to standard genetic algorithms [17] except it uses ruletrees instead of a bit representation. For a given transform and size, STEER generates a population of random ruletrees and times them. Through evolutionary techniques, STEER produces related new ruletrees and times them, searching for the fastest one. STEER times significantly less formulas than exhaustive search would but usually searches more of the formula space than dynamic programming.

These search algorithms must interface with the formula generator to produce the formulas that they wish to time. Ruletrees were specifically designed to be an efficient representation equivalent to a formula and a convenient interface between the search module and the formula generator. The search algorithms can very easily manipulate ruletrees without needing to parse through long formulas. Further, the search algorithms can interface with the formula generator to expand or change ruletrees as the search algorithms need. Dynamic programming needs the ability to apply all breakdown rules to any given transform and size, producing all possible sets of children for each applicable rule. A ruletree is a convenient data structure as dynamic programming will substitute for each of these children the ruletree that it has found to be fastest for that child's transform and size. STEER and random search requires the ability to choose a random applicable rule, and to choose randomly from its possible sets of children. For crossover, STEER takes advantage of the ruletree data structure to allow it to easily swap two subtrees between two ruletrees.

We conclude with a comparison of the different search strategies. Figure 5 shows the runtimes of the fastest formulas found by several search methods across several transforms. In general, STEER performs the best, outperforming DP for many of the transforms. However, STEER often times the most formulas; for example, DP times 156 formulas and STEER 1353 formulas for the DFT of size 2^{10}. We have also compared SPIRAL against FFTW 2.1.3. At size 2^4, FFTW is about 25% slower than SPIRAL probably due to the overhead caused by FFTW's plan data structure. Thus, we omitted this data point in the diagram. At size 2^{10}, SPIRAL is performing comparable with FFTW.

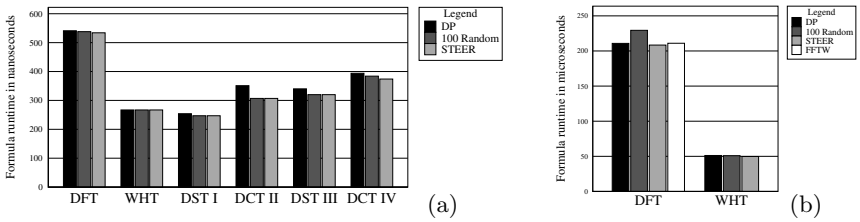


Fig. 5. Runtimes of the fastest formulas, implemented in C, found on a SUN UltraSparc-IIi 300 MHz for various transforms of size (a) 2^4 and (b) 2^{10}.

References

1. J. M. F. Moura, J. Johnson, R. W. Johnson, D. Padua, V. Prasanna, M. Püschel, and M. M. Veloso, SPIRAL: Portable Library of Optimized Signal Processing Algorithms, 1998, `http://www.ece.cmu.edu/~spiral`.
2. Matteo Frigo and Steven G. Johnson, FFTW: An adaptive software architecture for the FFT, in *ICASSP 98*, 1998, vol. 3, pp. 1381 1384, `http://www.fftw.org`.
3. C. Überhuber et.al., Aurora, `http://www.math.tuwien.ac.at/~aurora/`.
4. M. Thottethodi, S. Chatterjee, and A. R. Lebeck, Tuning Strassen's Matrix Multiplication for Memory E ciency, in *Proc. SC98: High Performance Networking and Computing*, 1998.
5. J. Demmel et.al., PHIPAC, `http://www.icsi.berkeley.edu/~bilmes/phipac/`.
6. R. C. Whaley, A. Petitet, and J. J. Dongarra, Automated Empirical Optimization of Software and the ATLAS project, Tech. Rep., University of Knoxville, Tennessee, 2000, `http://www.netlib.org/atlas/`.
7. J. Xiong, D. Padua, and J. Johnson, SPL: A Language and Compiler for DSP Algorithms, in *Proc. PLDI*, 2001, to appear.
8. The GAP Team, University of St. Andrews, Scotland, *GAP – Groups, Algorithms, and Programming*, 1997, `http://www-gap.dcs.st-and.ac.uk/~gap/`.
9. J. Johnson and R. W. Johnson, Automatic generation and implementation of FFT algorithms, in *Proc. SIAM Conf. Parallel Proc. for Sci. Comp.*, 1999, CD-Rom.
10. J. W. Cooley and J. W. Tukey, An algorithm for the machine calculation of complex Fourier series, *Math. of Computation*, vol. 19, pp. 297 301, 1965.
11. K. R. Rao and J. J. Hwang, *Techniques & standards for image, video and audio coding*, Prentice Hall PTR, 1996.
12. R. Tolimieri, M. An, and C. Lu, *Algorithms for discrete Fourier transforms and convolution*, Springer, 2nd edition, 1997.
13. Z. Wang, Fast Algorithms for the Discrete W Transform and for the Discrete Fourier Transform , *IEEE Trans. on Acoustics, Speech, and Signal Processing*, vol. ASSP-32, no. 4, pp. 803 816, 1984.
14. S. Egner and M. Püschel, *AREP – Constructive Representation Theory and Fast Signal Transforms*, GAP share package, 1998, `http://www.ece.cmu.edu/~smart/arep/arep.html`.
15. H. W. Johnson and C. S. Burrus, The design of optimal DFT algorithms using dynamic programming, *IEEE Trans, on Acoustics, Speech, and Signal Processing*, vol. ASSP-31, pp. 378 387, 1983.
16. B. Singer and M. Veloso, Stochastic search for signal processing algorithm optimization, in *Conf. on Uncertainty in Artificial Intelligence*, 2001, submitted.
17. David E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison-Wesley, Reading, MA, 1989.

Cache-Efficient Multigrid Algorithms*

Sriram Sellappa[1] and Siddhartha Chatterjee[2]

[1] Nexsi Corporation, 1959 Concourse Drive, San Jose, CA 95131
Email: sriram.sellappa@nexsi.com
[2] Department of Computer Science, The University of North Carolina
Chapel Hill, NC 27599-3175
Email: sc@cs.unc.edu

Abstract. Multigrid is widely used as an efficient solver for sparse linear systems arising from the discretization of elliptic boundary value problems. Linear relaxation methods like Gauss-Seidel and Red-Black Gauss-Seidel form the principal computational component of multigrid, and thus affect its efficiency. In the context of multigrid, these iterative solvers are executed for a small number of iterations (2–8). We exploit this property of the algorithm to develop a cache-efficient multigrid, by focusing on improving the memory behavior of the linear relaxation methods. The efficiency in our cache-efficient linear relaxation algorithm comes from two sources: reducing the number of data cache and TLB misses, and reducing the number of memory references by keeping values register-resident. Experiments on five modern computing platforms show a performance improvement of 1.15–2.7 times over a standard implementation of Full Multigrid V-Cycle.

1 Introduction

The growing speed gap between processor and memory has led to the development of memory hierarchies and to the widespread use of caches in modern processors. However, caches by themselves are not a panacea. Their success at reducing the average memory access time observed by a program depends on statistical properties of its dynamic memory access sequence. These properties generally go under the name of "locality of reference" and can by no means be assumed to exist in all codes. Compiler optimizations such as *iteration space tiling* [13,12] attempt to improve the locality of the memory reference stream by altering the schedule of program operations while preserving the dependences in the original program. While the theory of such loop transformations is well-developed, the choice of parameters remains a difficult optimization problem.

The importance of locality of reference is even more critical for hierarchical computations based on techniques such as multigrid, fast multipole, and wavelets, which

* This work was performed when the first author was a graduate student at UNC Chapel Hill. This work is supported in part by DARPA Grant DABT63-98-1-0001, NSF Grants EIA-97-26370 and CDA-95-12356, The University of North Carolina at Chapel Hill, Duke University, and an equipment donation through Intel Corporation's Technology for Education 2000 Program. The views and conclusions contained herein are those of the authors and should not be interpreted as representing the official policies or endorsements, either expressed or implied, of DARPA or the U.S. Government.

V.N. Alexandrov et al. (Eds.): ICCS 2001, LNCS 2073, pp. 107–116, 2001.
© Springer-Verlag Berlin Heidelberg 2001

typically perform $\Theta(1)$ operations on each data element. This is markedly different from dense matrix computations, which perform $\mathcal{O}(n^\epsilon)$ operations per data element (with $\epsilon > 0$) and can profit from data copying [7]. The lack of "algorithmic slack" in hierarchical codes makes it important to reduce both *the number of memory references* and *the number of cache misses* when optimizing them. Such optimizations can indeed be expressed as the combination of a number of standard compiler optimizations, but even the best current optimizing compilers are unable to synthesize such long chains of optimizations automatically. In this paper, we apply these ideas to develop cache-efficient multigrid.

The remainder of the paper is organized as follows. Section 2 introduces the problem domain. Section 3 discusses cache-efficient algorithms for this problem. Section 4 presents experimental results. Section 5 discusses related work. Section 6 summarizes.

2 Background

Many engineering applications involve boundary value problems that require solving elliptic differential equations. The discretization of such boundary value problems results in structured but sparse linear systems $Av = f$, where v is the set of unknowns corresponding to the unknown variables in the differential equation and f is the set of discrete values of the known function in the differential equation. A is a sparse matrix, whose structure and values depend on the parameters of discretization and the coefficients in the differential equation. Since A has few distinct terms, it is generally represented implicitly as a stencil kernel.

Such systems are often solved using iterative solvers such as linear relaxation methods, which naturally exploit the sparsity in the system. Each iteration of a linear relaxation method involves refining the current approximation to the solution by updating each element based on the approximation values at its neighbors. Figure 1 shows three common relaxation schemes: Jacobi, Gauss-Seidel, and Red-Black Gauss-Seidel. We consider a two-dimensional five-point kernel that arises, for example, from the discretization of Poisson's equation on the unit square. Of these, the Jacobi method is generally not used as a component of multigrid because of its slow convergence and its additional memory requirements. We therefore do not consider it further.

The error in the approximate solution can be decomposed into oscillatory and smooth components. Linear relaxation methods can rapidly eliminate the oscillatory components, but not the smooth components. For this reason, they are generally not used by themselves to solve linear systems, butt are used as building blocks for multigrid [3]. Multigrid improves convergence by using a hierarchy of successively coarser grids. In the multigrid context, linear relaxation methods are called *smoothers* and are run for a small number of iterations (2–8). We call this quantity `NITER`.

In addition to the smoother, multigrid employs *projection* and *interpolation* routines for transferring quantities between fine and coarse grids. Figure 2 shows the Full Multigrid V-cycle algorithm that we consider in this paper. Of these three components, the smoother dominates in terms of the number of computations and memory references. (For `NITER` =4, we have found it to take about 80% of total time.)

(a) Five-point Jacobi

```
for (m = 0; m < NITER; m++) {
 for (i = 1; i < (N-1); i++)
  for (j = 1; j < (N-1); j++)
   U[i,j] = w1*V[i,j-1] + w2*V[i-1,j] + w3*V[i,j] +
            w4*V[i+1,j] + w5*V[i,j+1] + w6*f[i,j];
 Swap(U,V);
}
```

(b) Five-point Gauss-Seidel

```
for (m = 0; m < NITER; m++)
 for (i = 1; i < (N-1); i++)
  for (j = 1; j < (N-1); j++)
   V[i,j] = w1*V[i,j-1] + w2*V[i-1,j] + w3*V[i,j] +
            w4*V[i+1,j] + w5*V[i,j+1] + w6*f[i,j];
```

(c) Five-point Red-Black Gauss-Seidel

```
for (m = 0; m < NITER; m++) {
  offset = 1;
  for (i = 1; i < (N-1); i++) {
    offset = 1-offset;
    for (j = 1+offset; j < (N-1); j += 2) {
      V[i,j] = w1*V[i,j-1] + w2*V[i-1,j] + w3*V[i,j] +
               w4*V[i+1,j] + w5*V[i,j+1] + w6*f[i,j];
    }
  }

  offset = 0;
  for (i = 1; i < (N-1); i++) {
    offset = 1-offset;
    for (j = 1+offset; (j) < (N-1); j += 2) {
      V[i,j] = w1*V[i,j-1] + w2*V[i-1,j] + w3*V[i,j] +
               w4*V[i+1,j] + w5*V[i,j+1] + w6*f[i,j];
    }
  }
}
```

Fig. 1. Code for three common linear relaxation methods.

$MV^h(v^h, f^h)$

1. Relax ν_1 times on $A^h u^h = f^h$ with initial guess v^h.
2. If $\Omega^h \neq$coarsest grid then
   $f^{2h} = Project(f^h - A^h v^h)$
   $v^{2h} = 0$
   $v^{2h} = MV^{2h}(v^{2h}, f^{2h})$
   $v^h = v^h + Interpolate(v^{2h})$.
3. Relax ν_2 times on $A^h u^h = f^h$ with initial guess v^h.
4. Return v^h.

(a) V-cycle Multigrid

{Initialize v^h,v^{2h},... to zero}

$FMV^h(v^h, f^h)$

1. If $\Omega^h \neq$coarsest grid then
   $f^{2h} = Project(f^h - A^h v^h)$
   $v^{2h} = 0$
   $v^{2h} = FMV^{2h}(v^{2h}, f^{2h})$
   $v^h = v^h + Interpolate(v^{2h})$.
2. $v^h = MV^h(v^h, f^h)$ ν_0 times.
   /* Invoke V-cycle ν_0 times to refine the solution */
3. Return v^h.

(b) Full Multigrid V-cycle

Fig. 2. Multigrid algorithms. Ω^h is a grid with grid spacing h. A superscript h on a quantity indicates that it is defined on Ω^h.

We now consider the memory system behavior of smoothers in terms of the 3C model [6] of cache misses.

- The classical Gauss-Seidel algorithm makes `NITER` sweeps over the whole array (2*`NITER` sweeps in the case of Red-Black Gauss-Seidel), accessing each element `NITER` times. Accesses to any individual element are temporally distant; since the array size is larger than the capacity of the cache, the element is likely to have been evicted from the cache before its access in the next iteration. The multiple sweeps of the array thus result in *capacity misses* in the data cache.
- The computation at an element in the array involves the values at the adjacent elements. So there is some spatial locality in the data. But the data dependences make it difficult for compilers to exploit this spatial locality.
- There could be *conflict misses* between the `V` and `f` arrays in Figure 1.
- The repetitive sweeps across the array cause address translation information to cycle through the (highly associative) TLB, which is deleterious to its performance. As the matrix dimension n grows, a virtual memory page will hold only $\Theta(1)$ rows or columns, requiring $\Theta(n)$ TLB entries to map the entire array. The resulting *capacity misses* in the TLB can be quite expensive given the high miss penalty.

The above observations motivate the algorithmic changes described in Section 3 that lead to cache-efficient multigrid algorithms.

3 Cache-Efficient Multigrid Algorithms

Our improvements to the efficiency of FMV stem exclusively from improvements to the memory behavior of the underlying smoothers. Two characteristics of these schemes are critical in developing their *cache-efficient* versions. First, we exploit the fact that the relaxation is run for a small number of iterations (2–8) by employing a form of iteration-space tiling [13] to eliminate the capacity misses incurred by the standard algorithm. Second, we exploit the spatial locality in the relaxation by retaining as many values in the registers as possible, using *stencil optimization* [4] to reduce the number of memory references. We describe our cache-efficient algorithms for two-dimensional, 5-pt Gauss-Seidel and Red-Black Gauss-Seidel schemes. We call these cache-efficient algorithms *temporal blocking* algorithms [2], because they partition the array into blocks and process blocks lexicographically to enhance temporal proximity among memory references. Note that these techniques preserve all data dependences of the standard (cache-unaware) algorithm. **Hence our cache-efficient algorithm is numerically identical to the standard algorithm.**

3.1 Cache-Efficient Gauss-Seidel Algorithm

The key idea in *temporal blocking* is to smoothen a subgrid of the solution matrix `NITER` times before moving on to the next subgrid; this clusters the `NITER` accesses to a particular element in time. We choose the subgrid size to fit in L1 cache; hence there are no capacity misses, as long as we touch only the elements within that subgrid, while working on that subgrid. Subgrids are square, of size $K * K$; boundary subgrids

are possibly rectangular. Gauss-Seidel requires elements to be updated in lexicographic order, requiring subgrids to also be visited the same way.

Consider the lowermost leftmost subgrid. All the elements of the subgrid can be updated once, except the elements at the right and top boundaries (to update them we need their neighbors, some of which lie outside the subgrid). Similarly, among the elements that were updated once, all the elements—except those on the right and top boundaries—can be updated again. Thus, for each additional iteration, the boundary of the elements with updated values shrinks by one along both dimensions. As a result, we have a *wavefront* of elements of width NITER that were updated from 1 to NITER-1 times. This wavefront propagates from the leftmost subgrid to the rightmost subgrid and is absorbed at the boundary of the matrix, through overlap between adjacent subgrids. Figure 3(b) shows the layout of overlapping subgrids, with NITER +1 rows and columns of overlap. The effect of NITER relaxation steps is illustrated for a subgrid in Figure 3(a) and for the entire matrix in Figure 4.

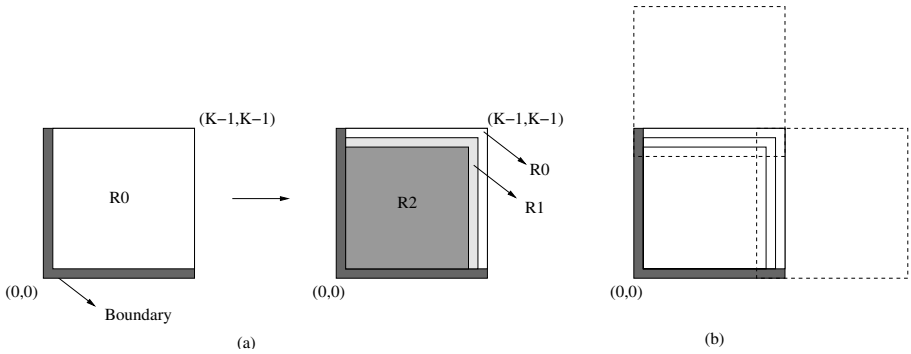


Fig. 3. (a) Transformation of the lowermost-leftmost subgrid by the *temporal blocking* algorithm for NITER =2. R0 is the set of elements that have not been updated, R1 is the set of elements that have been updated once, and R2 is the set of elements that have been updated twice. (b) The layout of the overlapping subgrids in the matrix.

The temporal blocked algorithm and the standard algorithm are numerically identical. The important performance difference between them comes from their usage of the memory system. Each subgrid is brought into the L1 cache once, so working within a subgrid does not result in capacity misses. There is some overlap among subgrids, and the overlapping regions along one dimension are fetched twice. Since NITER is 2–8, the overlapping region is small compared to the subgrid size, and the temporal blocking algorithm effectively makes a single pass over the array, *independent of NITER* . In contrast, the standard algorithm makes NITER passes over the array even if a compiler tiles the two innermost loops of Figure 1(b).

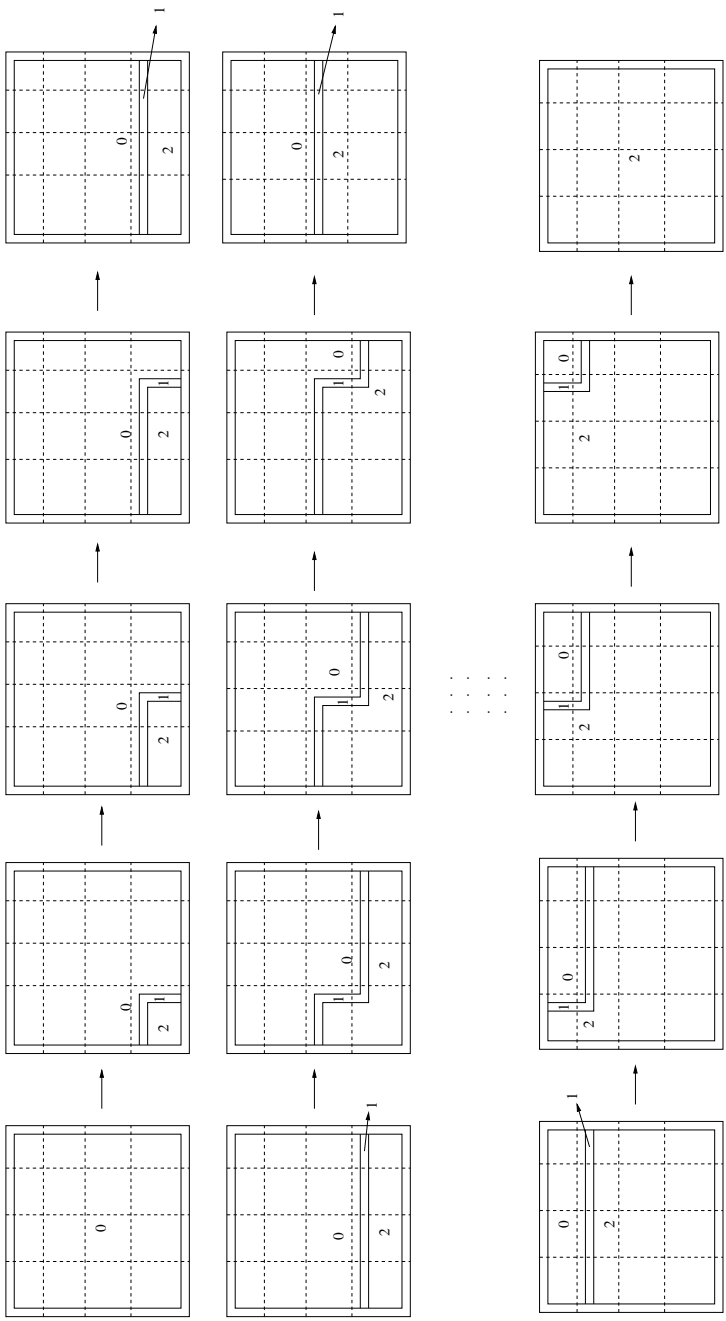


Fig. 4. Operation of the temporal blocking algorithm for Gauss-Seidel for `NITER` = 2. The initial matrix is the lowermost-leftmost matrix, and the final matrix is the rightmost-topmost matrix.

3.2 Stencil Optimization

Temporal blocking propagates the wavefront in a subgrid and pushes it to the beginning of the next subgrid. This shifting of the wavefront by one column at a time is a stencil operation where each element is updated using its neighbors and the elements are updated in lexicographic order. Each element of the subgrid is referenced five times in a single iteration of the `m`-loop in Figure 1(b): once for updating each of its four neighbors and once for updating itself. Note that, except for debugging situations, the intermediate values of the `V` array are not of interest; we care only about the final values of the elements after performing `NITER` steps of relaxation. This suggests that we might be able to read in each element value once, have it participate in multiple updates (to itself and to its neighbors) while remaining register-resident, and write out only the final updated value at the end of this process. If the value of `NITER` is small and the machine has enough floating-point registers, then this optimization is in fact feasible. What we have to do is to explicitly manage the registers as a small cache of intermediate results.

Performing stencil optimization at the source level requires care in programming (using explicit data transfers among several scalar variables) and availability of registers. Given the small value of `NITER`, the live variables fit within the register files available on most modern machines, and hence stencil optimization is very effective.

3.3 Cache-Efficient Red-Black Gauss-Seidel

Temporal blocking for Red-Black Gauss-Seidel is similar to that for Gauss-Seidel. The only difference is that the edges of the wavefront in this algorithm are sawtooth lines rather than straight lines, for the following reason. As we need the updated red elements to update the black elements, the boundary of the maximum number of elements that can be updated once is determined by the red elements in the subgrid, and the line joining the red elements has a sawtooth pattern. As a result, the width of the wavefront is 2*`NITER`. Other details of temporal blocking, like the propagation of the wavefront, remain unchanged. Stencil optimizations discussed above also apply in this case.

4 Experimental Results

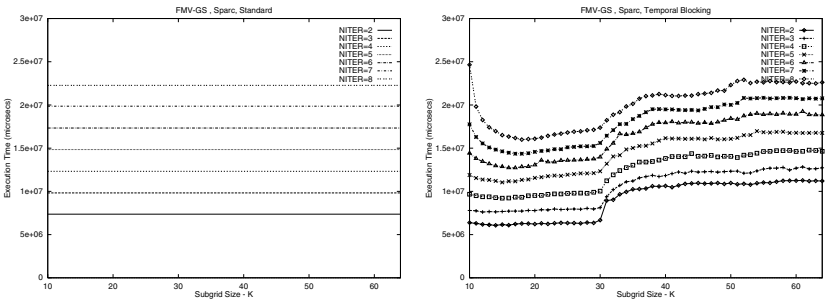
In this section we compare the performance of the standard and cache-efficient implementations of Full Multigrid V-cycle (FMV) with experimental results on a number of machines. We experimented on five commonly used modern computing platforms—UltraSPARC 60, SGI Origin 2000, AlphaPC 164LX, AlphaServer DS10, and Dell workstation—with both Gauss-Seidel and Red-Black Gauss-Seidel smoothers. Our test case is a two-dimensional Poisson's problem of size 1025×1025, with $\nu_0 = 4$ and $\nu_1 = \nu_2$ = `NITER` in Figure 2. The temporal blocking algorithm has one other parameter: K, the height of the subgrid. We are primarily interested in execution times of the algorithms. We use L1 cache misses, L2 cache misses, and TLB misses to explain the trends in execution time. Table 1 summarizes the overall performance improvement across platforms. For lack of space, we analyze the experimental data only for FMV with Gauss-Seidel relaxation on the Sparc.

Table 1. Ratio of running time of the standard version of FMV to the running time of the cache-efficient version, for Gauss-Seidel and Red-Black Gauss-Seidel relaxation schemes, on five modern computing platforms. The test problem is a two-dimensional Poisson problem of size 1025×1025. Larger numbers are better.

Platform	CPU	Clock speed	Gauss-Seidel	Red-Black Gauss-Seidel
UltraSPARC 60	UltraSPARC-II	300 MHz	1.35	2.4
SGI Origin 2000	MIPS R12000	300 MHz	1.35	2.4
AlphaPC 164LX	Alpha 21164	599 MHz	2.2	2.7
AlphaServer DS10	Alpha 21264	466 MHz	2.2	2
Dell Workstation	Pentium II	400 MHz	1.15	2

Figures 5(a) and (b) plot subgrid size *vs.* running time on the Sparc, one curve for each value of `NITER`. The plots demonstrate that the temporal blocking algorithm runs about 35% faster than the standard algorithm.

The plots in Figure 5(b) show an increase in running time of the cache-efficient FMV as the subgrid size increases, which is explained by TLB misses. All memory hierarchy simulations were performed using Lebeck's *fast-cache* and *cprof* simulators [8], for `NITER` = 4. Figure 6(a) shows the plot of TLB misses, which correlates with the degradation in running times for large subgrid sizes. The reason for the increase in the TLB misses is as follows. Since the size of the solution array is large, each row gets mapped to one or more virtual memory pages. When the temporal blocking algorithm works within a subgrid, the TLB needs to hold all the mapping entries of elements in that subgrid in the solution array (and the array of function values) in order to avoid additional TLB misses. Beyond a particular grid size, the number of TLB entries required exceeds the capacity of the TLB.



(a) (b)

Fig. 5. FMV with Gauss-Seidel relaxation, N=1025, and ν_0=4 on the Sparc. (a) Running time, standard version. (b) Running time, temporal blocked version.

Figure 6(b) shows the L1 cache misses on the Sparc. While the temporal blocking algorithm has fewer cache misses than the standard algorithm, the number of L1 cache misses increases with increase in subgrid size. Figures 6(c) and (d) show that conflict

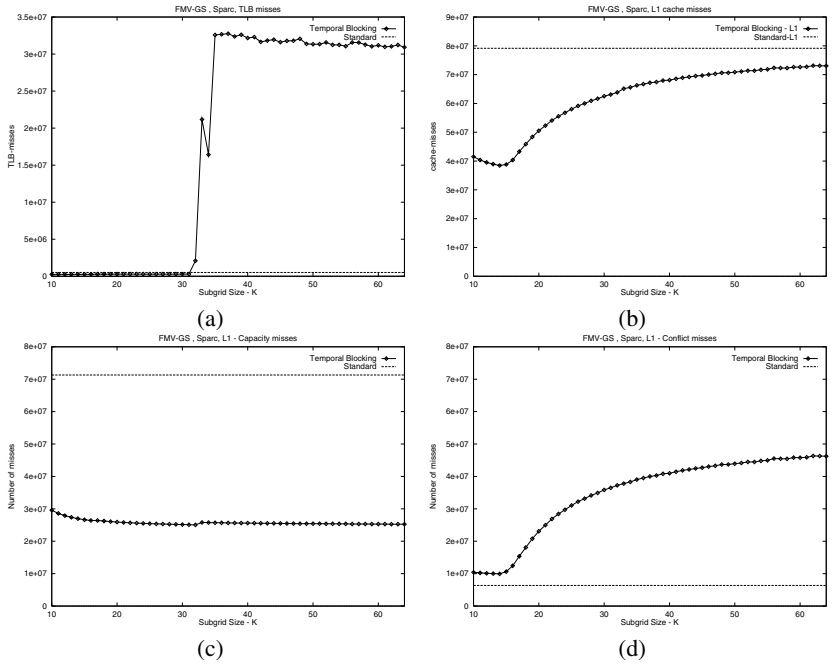


Fig. 6. FMV with Gauss-Seidel relaxation, N=1025, and ν_0=4 on the Sparc. (a) Number of TLB misses, `NITER` = 4. (b) Number of L1 cache misses, `NITER` = 4. (c) Number of L1 capacity misses, `NITER` = 4. (d) Number of L1 conflict misses, `NITER` = 4.

misses cause this increase. We confirmed that the conflict misses are due to *cross interference* between the V and f arrays, by running a cache simulation for a version of the code without the reference to f in the stencil. L1 cache misses remained constant in this simulation.

5 Related Work

Leiserson *et al.* [9] provide a graph-theoretic foundation for efficient linear relaxation algorithms using the idea of *blocking covers*. Their work, set in the context of out-of-core algorithms, attempts to reduce the number of I/O operations. Bassetti *et al.* [2] investigate stencil optimization techniques in a parallel object-oriented framework and introduce the notion of temporal blocking. In subsequent work [1], they integrate the blocking covers [9] work with their framework for the Jacobi scheme. Stals and Rüde [11] studied program transformations for the Red-Black Gauss-Seidel method. They explore blocking along one dimension for two-dimensional problems, but our work involves two-dimensional blocking. Douglas *et al.* [5] investigate cache optimizations for structured and unstructured multigrid. They focus only on the Red-Black Gauss-Seidel relaxation scheme, Povitsky [10] discusses a different wavefront approach to a cache-friendly algorithm to solve PDEs.

Bromley *et al.* [4] developed a compiler module to optimize stencil computations on the Connection Machine CM-2. To facilitate this, they worked with a particular style of specifying stencils in CM Fortran. They report performance of over 14 gigaflops. Their work focuses on optimizing a single application of a stencil, but does not handle the repeated application of a stencil that is characteristic of multigrid smoothers. Moreover, their technique does not handle cases when the stencil operations are performed in a non-simple order, like the order of updates in Red-Black Gauss-Seidel.

6 Conclusions

We have demonstrated improved running times for multigrid using a combination of algorithmic ideas, program transformations, and architectural capabilities. We have related these performance gains to improved memory system behavior of the new programs.

References

1. F. Bassetti, K. Davis, and M. Marathe. Improving cache utilization of linear relaxation methods: Theory and practice. In *Proceedings of ISCOPE'99*, Dec. 1999.
2. F. Bassetti, K. Davis, and D. Quinlan. Optimizing transformations of stencil operations for parallel object-oriented scientific frameworks on cache-based architectures. In *Proceedings of ISCOPE'98*, Dec. 1998.
3. W. L. Briggs. *A Multigrid Tutorial*. SIAM, 1987.
4. M. Bromley, S. Heller, T. McNerney, and G. L. Steele Jr. Fortran at ten gigaflops: The Connection Machine convolution compiler. In *Proceedings of the ACM SIGPLAN'91 Conference on Programming Language Design and Implementation*, pages 145–156, Toronto, Canada, June 1991.
5. C. Douglas, J. Hu, M. Kowarschik, U. Rüde, and C. Weiss. Cache optimization for structured and unstructured grid multigrid. *Electronic Transactions on Numerical Analysis*, 10:21–40, 2000. University of Kentucky, Louisville, KY, USA. ISSN 1068–9613.
6. M. D. Hill and A. J. Smith. Evaluating associativity in CPU caches. *IEEE Trans. Comput.*, C-38(12):1612–1630, Dec. 1989.
7. M. S. Lam, E. E. Rothberg, and M. E. Wolf. The cache performance and optimizations of blocked algorithms. In *Proceedings of the Fourth International Conference on Architectural Support for Programming Languages and Operating Systems*, pages 63–74, Apr. 1991.
8. A. R. Lebeck and D. A. Wood. Cache profiling and the SPEC benchmarks: A case study. *IEEE Computer*, 27(10):15–26, Oct. 1994.
9. C. E. Leiserson, S. Rao, and S. Toledo. Efficient out-of-core algorithms for linear relaxation using blocking covers. *J. Comput. Syst. Sci.*, 54(2):332–344, 1997.
10. A. Povitsky. Wavefront cache-friendly algorithm for compact numerical schemes. Technical Report 99-40, ICASE, Hampton, VA, Oct. 1999.
11. L. Stals and U. Rüde. Techniques for improving the data locality of iterative methods. Technical Report MRR 038-97, Institut für Mathematik, Universität Augsburg, Augsburg, Germany, Oct. 1997.
12. M. E. Wolf and M. S. Lam. A data locality optimizing algorithm. In *Proceedings of the ACM SIGPLAN'91 Conference on Programming Language Design and Implementation*, pages 30–44, Toronto, Canada, June 1991.
13. M. J. Wolfe. More iteration space tiling. In *Proceedings of Supercomputing'89*, pages 655–664, Reno, NV, Nov. 1989.

Statistical Models for Automatic Performance Tuning

Richard Vuduc[1], James W. Demmel[2], and Jeff Bilmes[3]

[1] Computer Science Division
University of California at Berkeley, Berkeley, CA 94720 USA
richie@cs.berkeley.edu
[2] Computer Science Division and Dept. of Mathematics
University of California at Berkeley, Berkeley, CA 94720 USA
demmel@cs.berkeley.edu
[3] Dept. of Electrical Engineering
University of Washington, Seattle, WA USA
bilmes@ee.washington.edu

Abstract. Achieving peak performance from library subroutines usually requires extensive, machine-dependent tuning by hand. Automatic tuning systems have emerged in response, and they typically operate, at compile-time, by (1) generating a large number of possible implementations of a subroutine, and (2) selecting a fast implementation by an exhaustive, empirical search. This paper applies statistical techniques to exploit the large amount of performance data collected during the search. First, we develop a heuristic for stopping an exhaustive compile-time search early if a near-optimal implementation is found. Second, we show how to construct run-time decision rules, based on run-time inputs, for selecting from among a subset of the best implementations. We apply our methods to actual performance data collected by the PHiPAC tuning system for matrix multiply on a variety of hardware platforms.

1 Introduction

Standard library interfaces have enabled the development of portable applications that can also achieve *portable performance*, provided that optimized libraries are available and affordable on all platforms of interest to users. Example libraries in scientific applications include the Basic Linear Algebra Subroutines (BLAS) [11,5], the Vector and Signal Image Processing Library API [12], and the Message Passing Interface (MPI) for distributed parallel communications.

However, both construction and machine-specific hand-tuning of these libraries can be tedious and time-consuming tasks. Thus, several recent research efforts are automating the process using the following two-step method. First, rather than code particular routines by hand for each computing platform of interest, these systems contain parameterized code generators that encapsulate possible tuning strategies. Second, the systems tune for a particular platform by *searching*, i.e., varying the generators' parameters, benchmarking the resulting routines, and selecting the fastest implementation.

V.N. Alexandrov et al. (Eds.): ICCS 2001, LNCS 2073, pp. 117–126, 2001.
© Springer-Verlag Berlin Heidelberg 2001

In this paper, we focus on the possible uses of performance data collected during the search task.[1] Specifically, we first justify the need for exhaustive searches in Section 2, using actual data collected from an automatic tuning system. However, users of such systems cannot always afford to perform these searches. Therefore, we discuss a statistical model of the feedback data that allows users to stop the search early based on meaningful information about the search's progress in Section 3. Of course, a single implementation is not necessarily the fastest possible for all possible inputs. Thus, we discuss additional performance modeling techniques in Section 4 that allow us to select at run-time an implementation believed to perform best on a particular input. We apply these techniques to data collected from the PHiPAC system (see Section 2) which generates highly tuned matrix multiply implementations [1,2].

There are presently a number of other similar and important tuning systems. These include FFTW for discrete Fourier transforms [6], ATLAS [17] for the BLAS, Sparsity [8] for sparse matrix-vector multiply, and SPIRAL [7,13] for signal and image processing. Vadhiyar, et al. [14], explore automatically tuning MPI collective operations. These systems employ a variety of sophisticated code generators that use both the mathematical structure of the problems they solve and the characteristics of the underlying machine to generate high performance code. All match hand-tuned vendor libraries, when available, on a wide variety of platforms. Nevertheless, these systems also face the common problem of how to reduce the lengthy search process. Each uses properties specific to their code generators to prune the search spaces. Here, we present complementary techniques for pruning the search spaces independently of the code generator.

The search task deserves attention not only because of its central role in specialized tuning systems, but also because of its potential utility in compilers. Researchers in the OCEANS project [10] are integrating such an empirical search procedure into a general purpose compiler. Search-directed compilation should be valuable when performance models fail to charaterize source code adequately.

2 The Case for Searching

In this section, we present data to motivate the need for search methods in automated tuning systems, using PHiPAC as a case study. PHiPAC searches a combinatorially large space defined by possible optimizations in building its implementation. Among the most important optimizations are (1) register, L1, and L2 cache tile sizes where non-square shapes are allowed, (2) loop unrolling, and (3) a choice of six software pipelining strategies. To limit search time, machine parameters (such as the number of registers available and cache sizes) are used to restrict tile sizes. In spite of this and other pruning heuristics, searches generally can take hours to *weeks* depending on the user-selectable thoroughness of the search. Nevertheless, Figure 1 shows two examples in which the performance of PHiPAC-generated routines compares well with (a) hand-tuned vendor libraries and (b) "naive" C code (3-nested loops) compiled with full optimizations.

[1] An extended version of this paper has appeared elsewhere [16].

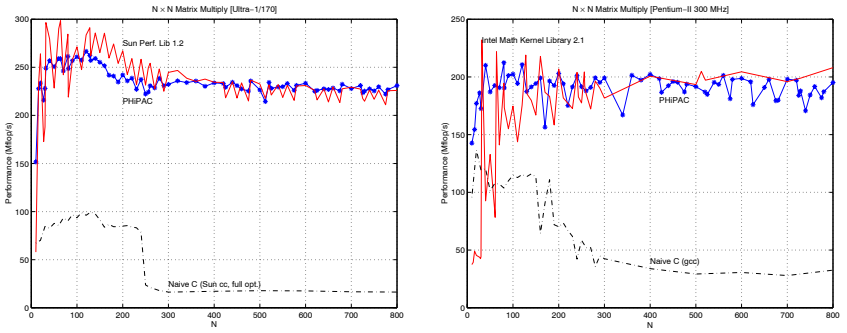


Fig. 1. Performance (Mflop/s) on a square matrix multiply benchmark for the Sun Ultra 1/170 workstation (*left*) and a 300 MHz Pentium-II platform (*right*). The theoretical peaks are 333 Mflop/s and 300 Mflop/s, respectively.

Exhaustive searches are often necessary to find the very best implementations, although a partial search can find near-optimal implementations. In an experiment we fixed a particular software pipelining strategy and explored the space of possible register tile sizes on six different platforms. This space is three-dimensional and we index it by integer triplets (m_0, k_0, n_0).[2] Using heuristics, this space was pruned to contain between 500 and 2500 reasonable implementations per platform. Figure 2 (*left*) shows what fraction of implementations (y-axis) achieved what fraction of machine peak (x-axis). On the IBM RS/6000, 5% of the implementations achieved at least 90% of the machine peak. By contrast, only 1.7% on a uniprocessor Cray T3E node, 4% on a Pentium-II, and 6.5% on a Sun Ultra1/170 achieved more than 60% of machine peak. And on a majority of the platforms, fewer than 1% of implemenations were within 5% of the best; 80% on the Cray T3E ran at less than 15% of machine peak. Two important ideas emerge: (1) different machines can display widely different characteristics, making generalization of search properties across them difficult, and (2) finding the very best implementations is akin to finding a "needle in a haystack."

The latter difficulty is illustrated in Figure 2 (*right*), which shows a 2-D slice ($k_0 = 1$) of the 3-D tile space on the Ultra. The plot is color coded from black=50 Mflop/s to white=270 Mflop/s. The lone white square at $(m_0 = 2, n_0 = 8)$ was the fastest. The black region to the upper-right was pruned (i.e., not searched) based on the number of registers. We see that performance is not a smooth function of algorithmic details, making accurate sampling and interpolation of the space difficult. Like Figure 2 (*left*), this motivates an exhaustive search.

3 Early Stopping Criteria

Unfortunately, exhaustive searches can be demanding, requiring dedicated machine time for long periods. Thus, tuning systems prune the search spaces using

[2] The specifics of why the space is three dimensional are, for the moment, unimportant.

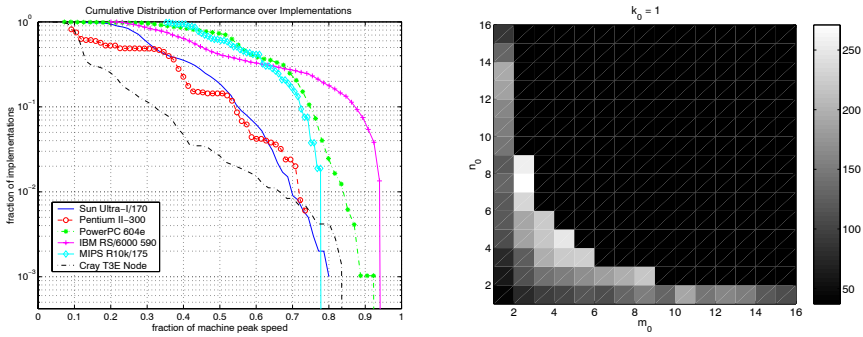


Fig. 2. (*Left*) The fraction of implementations (y-axis) attaining at least a given level of peak machine speed (x-axis) on six platforms. (*Right*) A 2-D slice of the 3-D register tile space on the Sun Ultra1/170 platform. The best implementation ($m_0 = 2, n_0 = 8$) achieved 271 Mflop/s.

application-specific heuristics. We consider a complementary method for stopping a search early based only on performance data gathered during the search.

More formally, suppose there are N possible implementations. When we generate implementation i, we measure its performance x_i. Assume that each x_i is normalized to lie between 0 (slowest) and 1 (fastest). Define the space of implementations as $S = \{x_1, \ldots, x_N\}$. Let X be a random variable corresponding to the value of an element drawn uniformly at random from S, and let $n(x)$ be the number of elements of S less than or equal to x. Then X has a cumulative distribution function (cdf) $F(x) = Pr[X \leq x] = n(x)/N$. At time t, where t is between 1 and N inclusive, suppose we generate an implementation at random *without* replacement. Let X_t be a random variable corresponding to the observed performance. Letting $M_t = \max_{1 \leq i \leq t} X_i$ be the maximum observed performance at t, we can ask about the chance that M_t is less than some threshold:

$$Pr[M_t \leq 1 - \epsilon] < \alpha, \tag{1}$$

where ϵ is the proximity to the best performance, and α is an upper-bound on the probability that the observed maximum at time t is below $1 - \epsilon$. Note that

$$Pr[M_t \leq x] = Pr[X_1 \leq x, X_2 \leq x, \ldots, X_t \leq x] = p_1(x)p_2(x)\cdots p_t(x) \tag{2}$$

where, assuming no replacement,

$$\begin{aligned} p_r(x) &= Pr[X_r \leq x | X_1 \leq x, \ldots, X_{r-1} \leq x] \\ &= \begin{cases} 0 & n(x) < r \\ \frac{n(x)-r+1}{N-r+1} & n(x) \geq r \end{cases} \end{aligned} \tag{3}$$

Since $n(x) = N \cdot F(x)$, we cannot know its true value since we do not know the true distribution $F(x)$. However we can use the t observed samples to approximate $F(x)$ using, say, the empirical cdf (ecdf) $\hat{F}_t(x)$ based on the t samples:

$$\hat{F}_t(x) = \hat{n}_t(x)/t \tag{4}$$

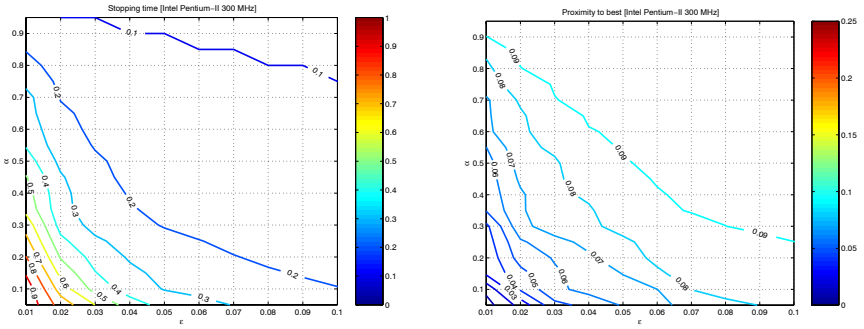


Fig. 3. Average stopping time (left), as a fraction of the total search space, and proximity to the best performance (right), as the difference between normalized performance scores, on the 300 MHz Pentium-II class workstation as functions of the tolerance parameters ϵ (x-axis) and α (y-axis). Note that the values shown are mean *plus* standard deviation, to give an approximate upper-bound on the average case.

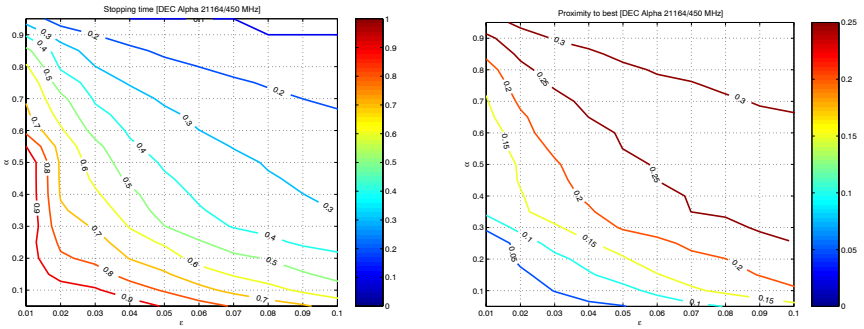


Fig. 4. Same as Figure 3 for a uniprocessor Cray T3E node.

where $\hat{n}_t(x)$ is the number of observed samples less than or equal to x. We *rescale* the samples so that the maximum is one, since we do not know the true maximum.[3] Other forms for equation (4) are opportunities for experimentation.

In summary, a user or library designer specifies the search tolerance parameters ϵ and α. Then at each time t, the automated search system builds the ecdf in equation (4) to estimate (2). The search ends when equation (1) is satisfied.

We apply the above model to the register tile space data to the platforms shown in Figure 2 (*left*). The results appear in Figures 3 and 4 for the Pentium and Cray T3E platforms, respectively. The left plots show the average stopping time *plus* the standard deviation as a function of ϵ and α; this gives a pessimistic bound on the average value. The right plots show the average proximity of the

[3] This was a reasonable approximation on actual data. We are developing theoretical bounds on the quality of this approximation, which we expect will be close to the known bounds on ecdf approximation due to Kolmogorov and Smirnov [3].

implementation found to the best one (again, plus the standard deviation), as a fraction. On the Pentium (Figure 3), setting $\epsilon = .05$ and $\alpha = .1$ we see that the search ends after sampling less than a third of the full space (left plot), having found an implementation within about 6.5% of the best (right plot). On the Cray T3E (Figure 4) where the best is difficult to find, the same tolerance values produce an implementation within about 8% of the best while still requiring exploration of 80% of the search space. Thus, the model adapts to the characteristics of the implementations and the underlying machine.

In prior work [1], we experimented with search methods including random, ordered, best-first, simulated annealing. The OCEANS project [10] has also reported on a quantitative comparison of these methods and others. In both, random search was comparable to and easier to implement than the others. Our technique adds user-interpretable bounds to the simple random method. Note that if the user wishes to specify a maximum search time (e.g., "stop searching after 3 hours"), the bounds could be computed and reported to the user.

4 Run-Time Selection Rules

The previous sections assume that a single, optimal implementation can be found. For some applications, however, several implementations may be "optimal" depending on the input parameters. Thus, we may wish to build decision rules to select an appropriate implementation based on the run-time inputs.

Formally, we want to solve the following problem. Suppose we are given (1) a set of m "good" implementations of an algorithm, $A = \{a_1, \ldots, a_m\}$ which all give the same output when presented with the same input; (2) a set of samples $S_0 = \{s_1, s_2, \ldots, s_n\}$ from the space S of all possible inputs (i.e., $S_0 \subseteq S$), where each s_i is a d-dimensional real vector; (3) the execution time $T(a, s)$ of algorithm a on input s, where $a \in A$ and $s \in S$. Our goal is to find a decision function $f(s)$ that maps an input s to the best implementation in A, i.e., $f : S \to A$. The idea is to construct $f(s)$ using the performance of the good implementations on a sample of the inputs S_0. We will refer to S_0 as the *training set*. In geometric terms, we would like to partition the input space by implementation. This would occur at compile (or "build") time. At run-time, the user calls a single routine which, when given an input s, evaluates $f(s)$ to select and execute an implementation.

There are a number of important issues. Among them is the cost and complexity of building f. Another is the cost of evaluating $f(s)$; this should be a fraction of the cost of executing the best implementation. A third issue is how to compare the prediction accuracy of different decision functions. One possible metric is the average misclassification rate, or fraction of test samples mispredicted (call it Δ_{miss}). We always choose the *test set* S' to exclude the training data S_0, that is, $S' \subseteq (S - S_0)$. However, if the performance difference between two implementations is small, a misprediction may still be acceptable. Thus, we also use the average slow-down of the selected variant relative to the best, Δ_{err}.

For example, consider the matrix multiply operation $C = C + AB$, where A, B, and C are dense matrices of size $M \times K$, $K \times N$, and $M \times N$, respectively. In

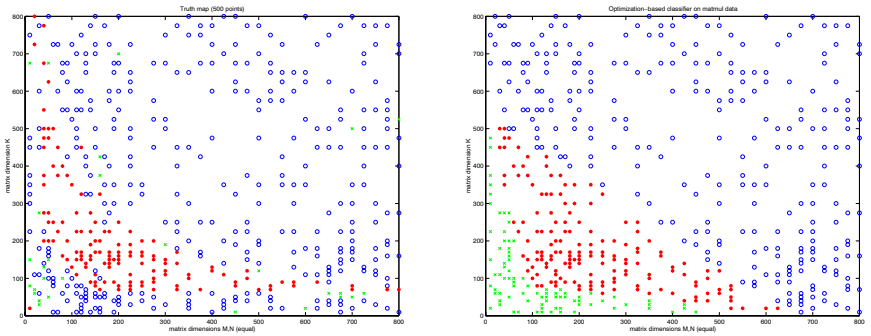


Fig. 5. (*Left*) A "truth map" showing the regions in which particular implementations are fastest. A 500-point sample of a 2-D slice of the input space is shown. Red *'s correspond to an implementation with only register tiling, green x's have L1 cache tiling, and blue o's have L1 and L2 tiling. (*Right*) Prediction results for the cost-based method.

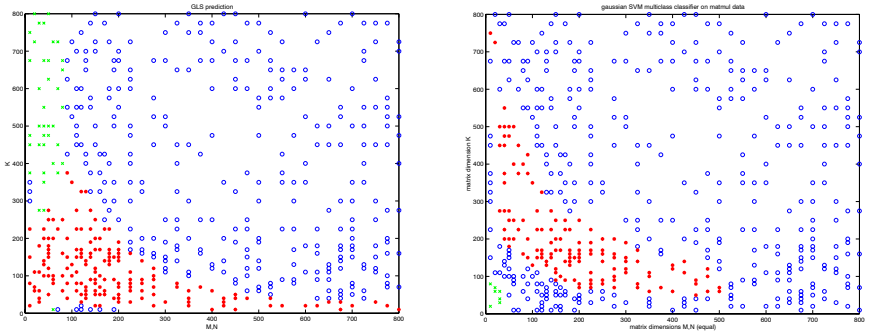


Fig. 6. Prediction results for the regression (*left*) and support-vector (*right*) methods.

PHiPAC, it is possible to generate different implementations tuned on different matrix workloads. For instance, we could have three implementations, tuned for matrix sizes that fit approximately within L1 cache, those that fit within L2, and all larger sizes. The inputs to each are M, K, and N, making the input space S three-dimensional. We will refer to this example in the following sections.

4.1 A Cost Minimization Method

Associate with each implementation a a weight function $w_{\theta_a}(s)$, parameterized by θ_a, which returns a value between 0 and 1 for some input value s. Our decision function selects the algorithm with the highest weight on input s, $f(s) = \text{argmax}_{a \in A} \{w_{\theta_a}(s)\}$. Compute the weights so as to minimize the average execution time over the training set, i.e., minimize

$$C(\theta_{a_1}, \ldots, \theta_{a_m}) = \sum_{a \in A} \sum_{s \in S_0} w_{\theta_a}(s) \cdot T(a, s). \tag{5}$$

Of the many possible choices for w_{θ_a}, we choose the *softmax function* [9], $w_{\theta_a}(s) = exp\left(\theta_a^T s + \theta_{a,0}\right)/Z$ where θ_a has the same dimensions as s, $\theta_{a,0}$ is an additional parameter to estimate, and Z is a normalizing constant. It turns out that the derivatives of the weights are easy to compute, so we can estimate θ_a and $\theta_{a,0}$ by minimizing equation (5) numerically using Newton's method. A nice property of the weight function is that f becomes cheap to evaluate at run-time.

4.2 Regression Models

Another natural idea is to postulate a parametric model for the running time of each implementation. Then at run-time, we can choose the fastest implementation based on the execution time predicted by the models. This approach was originally proposed by Brewer [4]. For matrix multiply on matrices of size $N \times N$, we might guess that the running time of implementation a will have the form

$$T_a(N) = \beta_3 N^3 + \beta_2 N^2 + \beta_1 N + \beta_0. \tag{6}$$

Given sample running times on some inputs S_0, we can use standard regression techniques to determine the β_k coefficients. The decision function is just $f(s) = \text{argmin}_{a \in A} T_a(s)$. An advantage of this approach is that the models, and thus the accuracy of prediction as well as the cost of making a prediction, can be as simple or as complicated as desired. Also, no assumptions are being made about the geometry of the input space, as with our cost-minimization technique. However, a difficult disadvantage is that it may not be easy to postulate an accurate run-time model.

4.3 The Support Vector Method

Another approach is to view the problem as a statistical classification task. One sophisticated and successful classification algorithm is known as the support vector (SV) method [15]. In this method, each sample $s_i \in S_0$ is given a label $l_i \in A$ to indicate which implementation was fastest for that input. The SV method then computes a partitioning that attempts to maximize the minimum distance between classes.[4] The result is a decision function $f(s)$. The SV method is reasonably well-grounded theoretically and potentially much more accurate than the previous two methods, and we include it in our discussion as a kind of practical upper-bound on prediction accuracy. However, the time to compute $f(s)$ is a factor of $|S_0|$ greater than that of the other methods and is thus possibly much more expensive to calculate at run-time.

4.4 Results with PHiPAC Data

We offer a brief comparison of the three methods on the matrix multiply example described previously. The predictions of the three methods on a sample test set

[4] Formally, this is the *optimal margin* criterion [15].

Table 1. The three predictors on matrix multiply. "Best 5%" is the fraction of predicted implementations whose execution times were within 5% of the best possible. "Worst 20%" and "50%" are the fraction less than 20% and 50% of optimal, respectively.

			Best	Worst	
Method	Δ_{miss}	Δ_{err}	*5%*	*20%*	*50%*
Regression	34.5%	2.6%	90.7%	1.2%	0.4%
Cost-Min	31.6%	2.2%	94.5%	2.8%	1.2%
SVM	12.0%	1.5%	99.0%	0.4%	0%

are shown in Figures 5 (right) and 6. Qualitatively, we see that the boundaries of the cost-based method are a poor fit to the data. The regression method captures the boundaries roughly but does not correctly model one of the implementations (upper-left of figure). The SV method appears to produce the best predictions.

Table 1 compares the accuracy of the three methods by the two metrics Δ_{miss} and Δ_{err}; in addition we report the fraction of test points predicted *within* 5% of the best possible, and the fraction predicted that were 20% and 50% *below* optimal. These values are averaged over ten training and test sets. The values for Δ_{miss} confirm the qualitative results shown in the figures. However, the methods are largely comparable by the Δ_{err} metric, showing that a high misclassification rate did not necessarily lead to poor performance overall. Note that the worst 20% and 50% numbers show that the regression method made slightly better mispredictions on average than the cost-minimization method. In addition, both the regression and cost-minimization methods lead to reasonably fast predictors. Prediction times were roughly equivalent to the execution time of a 3x3 matrix multiply. By contrast, the prediction cost of the SVM is about a 64x64 matrix multiply, which may prohibit its use when small sizes occur often.

However, this analysis is not intended to be definitive. For instance, we cannot fairly report on specific training costs due to differences in the implementations in our experimental setting. Also, matrix multiply is only one possible application. Instead, our aim is simply to present the general framework and illustrate the issues on actual data. Moreover, there are many possible models; our examples offer a flavor for the role that statistical modeling of performance data can play.

5 Conclusions and Directions

While all of the existing automatic tuning systems implicitly follow the two-step "generate-and-search" methodology, one aim of this study is to draw attention to the process of searching itself as an interesting and challenging problem.

One challenge is pruning the enormous implementation spaces. Existing tuning systems have shown the effectiveness of pruning these spaces using problem-specific heuristics; our black-box pruning method for stopping the search process early is a complementary technique. It has the nice properties of (1) incorporating performance feedback data, and (2) providing users with a meaningful way (namely, via probabilistic thresholds) to control the search procedure.

The other challenge is to find efficient ways to select implementations at run-time when several known implementations are available. Our aim has been to discuss a possible framework, using sampling and statistical classification, for attacking this problem in the context of automatic tuning systems. This connects high performance software engineering with statistical modeling ideas. Other modeling techniques and applications remain to be explored.

Acknowledgements. We wish to thank Andrew Ng for his feedback on our statistical methodology.

References

1. J. Bilmes, K. Asanović, C. Chin, and J. Demmel. Optimizing matrix multiply using PHiPAC: a Portable, High-Performance, ANSI C coding methodology. In *Proc. of the Int'l Conf. on Supercomputing, Vienna, Austria*, July 1997.
2. J. Bilmes, K. Asanović, J. Demmel, D. Lam, and C. Chin. The PHiPAC v1.0 matrix-multiply distribution. Technical Report UCB/CSD-98-1020, University of California, Berkeley, October 1998.
3. Z. W. Birnbaum. Numerical tabulation of the distribution of Kolmogorov's statistic for finite sample size. *J. Am. Stat. Assoc.*, 47:425–441, September 1952.
4. E. Brewer. High-level optimization via automated statistical modeling. In *Sym. Par. Alg. Arch., Santa Barbara, California*, July 1995.
5. J. Dongarra, J. D. Croz, I. Duff, and S. Hammarling. A set of level 3 basic linear algebra subprograms. *ACM Trans. Math. Soft.*, 16(1):1–17, March 1990.
6. M. Frigo and S. Johnson. FFTW: An adaptive software architecture for the FFT. In *Proc. of the Int'l Conf. on Acoustics, Speech, and Signal Processing*, May 1998.
7. G. Haentjens. An investigation of recursive FFT implementations. Master's thesis, Carnegie Mellon University, 2000.
8. E.-J. Im and K. Yelick. Optimizing sparse matrix vector multiplication on SMPs. In *Proc. of the 9th SIAM Conf. on Parallel Processing for Sci. Comp.*, March 1999.
9. M. I. Jordan. Why the logistic function? Technical Report 9503, MIT, 1995.
10. T. Kisuki, P. M. Knijnenburg, M. F. O'Boyle, and H. Wijshoff. Iterative compilation in program optimization. In *Proceedings of the 8th International Workshop on Compilers for Parallel Computers*, pages 35–44, 2000.
11. C. Lawson, R. Hanson, D. Kincaid, and F. Krogh. Basic linear algebra subprograms for Fortran usage. *ACM Trans. Math. Soft.*, 5:308–323, 1979.
12. D. A. Schwartz, R. R. Judd, W. J. Harrod, and D. P. Manley. VSIPL 1.0 API, March 2000. `www.vsipl.org`.
13. B. Singer and M. Veloso. Learning to predict performance from formula modeling and training data. In *Proc. of the 17th Int'l Conf. on Mach. Learn.*, 2000.
14. S. S. Vadhiyar, G. E. Fagg, and J. Dongarra. Automatically tuned collective operations. In *Proceedings of Supercomputing 2000*, November 2000.
15. V. N. Vapnik. *Statistical Learning Theory.* John Wiley and Sons, Inc., 1998.
16. R. Vuduc, J. Demmel, and J. Bilmes. Statistical modeling of feedback data in an automatic tuning system. In *MICRO-33: Third ACM Workshop on Feedback-Directed Dynamic Optimization*, December 2000.
17. C. Whaley and J. Dongarra. Automatically tuned linear algebra software. In *Proc. of Supercomp.*, 1998.

Optimizing Sparse Matrix Computations for Register Reuse in SPARSITY

Eun-Jin Im[1] and Katherine Yelick[2]

[1] School of Computer Science, Kookmin University, Seoul, Korea
ejim@cs.kookmin.ac.kr,
[2] Computer Science Division, University of California, Berkeley, CA 94720, USA
yelick@cs.Berkeley.EDU

Abstract. Sparse matrix-vector multiplication is an important computational kernel that tends to perform poorly on modern processors, largely because of its high ratio of memory operations to arithmetic operations. Optimizing this algorithm is difficult, both because of the complexity of memory systems and because the performance is highly dependent on the nonzero structure of the matrix. The Sparsity system is designed to address these problem by allowing users to automatically build sparse matrix kernels that are tuned to their matrices and machines. The most difficult aspect of optimizing these algorithms is selecting among a large set of possible transformations and choosing parameters, such as block size. In this paper we discuss the optimization of two operations: a sparse matrix times a dense vector and a sparse matrix times a set of dense vectors. Our experience indicates that for matrices arising in scientific simulations, register level optimizations are critical, and we focus here on the optimizations and parameter selection techniques used in Sparsity for register-level optimizations. We demonstrate speedups of up to 2× for the single vector case and 5× for the multiple vector case.

1 Introduction

Matrix-vector multiplication is used in scientific computation, signal and image processing, document retrieval, and many other applications. In many cases, the matrices are sparse, so only the nonzero elements and their indices are stored. The performance of sparse matrix operations tends to be much lower than their dense matrix counterparts for two reasons: 1) there is overhead to accessing the index information in the matrix structure and 2) the memory accesses tend to have little spatial or temporal locality. For example, on an 167 MHz UltraSPARC I, there is a 2x slowdown due to the data structure overhead (measured by comparing a dense matrix in sparse and dense format) and an additional 5x slowdown for matrices that have a nearly random nonzero structure.

The Sparsity system is designed to help users obtain highly tuned sparse matrix kernels without having to know the details of their machine's memory hierarchy or how their particular matrix structure will be mapped onto that hierarchy. Sparsity performs several optimizations, including register blocking, cache blocking, loop unrolling, matrix reordering, and reorganization for multiple

V.N. Alexandrov et al. (Eds.): ICCS 2001, LNCS 2073, pp. 127–136, 2001.
© Springer-Verlag Berlin Heidelberg 2001

vectors [Im00]. The optimizations involve both code and data structure transformations, which can be quite expensive. Fortunately, sparse matrix-vector multiplication is often used in iterative solvers or other settings where the same matrix is multiplied by several different vectors, or matrices with different numerical entries but the same or similar nonzero patterns will be re-used. Sparsity therefore uses transformations that are specialized to a particular matrix structure, which we will show is critical to obtaining high performance.

In this paper we focus on register level optimizations, which include register blocking and reorganization for multiple vectors. The challenge is to select the proper block size and the right number of vectors to maximize performance. In both cases there are trade-offs which make the parameters selection very sensitive to both machine and matrix. We explore a large space of possible techniques, including searching over a set of parameters on the machine and matrix of interest and use of performance models to predict which parameter settings will perform well. For setting the register block size, we present a performance model based on some matrix-independent machine characteristics, combined with an analysis of blocking factors that is computed by a statistical sampling of the matrix structure. The model works well in practice and eliminates the need for a large search. For choosing the optimal number of vectors in applications where a large number or vectors are used, we present a heuristic for choosing the block size automatically, which works well on many matrices, but in some cases we find that searching over a small number of vectors produces much better results.

2 Register Optimizations for Sparse Matrices

In this section we describe two optimizations: register blocking and reorganization for multiple vectors. There are many popular sparse matrix formats, but to make this discussion concrete, assume we start with a matrix in Compressed Sparse Row (CSR) format. In CSR, all row indices are stored (by row) in one vector, all matrix values are stored in another, and a separate vector of indices indicates where each row starts within these two vectors. In the calculation of $y = A \times x$, where A is a sparse matrix and x and y are dense vectors, the computation may be organized as a series of dot-products on the rows. In this case, the elements of A are accessed sequentially but not reused. The elements of y are also accessed sequentially, but more importantly they are re-used for each nonzero in the row of A. The access to x is irregular, as it depends on the column indices of nonzero elements in matrix A.

Register re-use of y and A cannot be improved, but access to x may be optimized if there are elements in A that are in the same column and nearby one another, so that an element of x may be saved in a register. To improve locality, Sparsity stores a matrix as a sequence of small dense blocks, and organizes the computation to compute each block before moving on to the next. To take advantage of the improved locality for register allocation, the block sizes need to be fixed at compile time. Sparsity therefore generates code for matrices containing only full dense blocks of some fixed size $r \times c$, where each block starts on

a row that is a multiple of r and a column that is a multiple of c. The code for each block is also unrolled, with instruction scheduling and other optimizations applied by the C compiler. The assumption is that all nonzeros must be part of some $r \times c$ block, so Sparsity will transform the data structure to add explicit zeros where necessary. While the idea of *blocking* or *tiling* for dense matrix operations is well-known (e.g., [LRW91]), the sparse matrix transformation is quite different, since it involves filling in zeros, and the choice of r and c will depend on the matrix structure as described in section 3.

We also consider a second register level optimization of matrix-vector multiplication when the matrix is going to be multiplied by a set of vectors. This is less common than the single vector case, but practical when there are multiple right-hand sides in an iterative solver, or in blocked eigenvalue algorithms, such as block Lanczos [Mar95] or block Arnoldi [BCD$^+$00]. Matrix-vector multiplication accesses each matrix element only once, whereas a matrix times a set of k vectors will access each matrix element k times. While there is much more potential for high performance with multiple vectors, the advantage will not be exhibited in straightforward implementations. The basic optimization is to interchange loops so that for each matrix element, the source and destination values for all vectors are accessed before going to the next element.

Sparsity contains a code generator that produces loop-unrolled C code for given block sizes and for a fixed number of vectors. If the number of vectors is very large, the loop over the vectors is strip-mined, with the resulting inner loop becoming one of these unrolled loops. The optimized code removes some of the branch statements and load stalls by reordering instructions, all of which further improve the performance beyond simply interchanging loops.

3 Choosing the Register Block Size

Register blocking does not always improve performance if the sparse matrix does not have small dense blocks. Even when it has such blocks, the optimizer must pick a good block size for a given matrix and machine. We have developed a performance model that predicts the performance of the multiplication for various block sizes without actually blocking and running the multiplication. The model is used to select a good block size.

There is a trade-off in the choice of block size for sparse matrices. In general, the computation rate will increase with the block size, up to some limit at which register spilling becomes necessary. In most sparse matrices, the dense sub-blocks that arise naturally are relatively small: 2×2, 3×3 and 6×6 are typical values. When a matrix is converted to a blocked format, some zero elements are filled in to make a complete $r \times c$ block. These extra zero values not only consume storage, but increase the number of floating point operations, because they are involved in the sparse matrix computation. The number of added zeros in the blocked representation are referred to as *fill*, and the ratio of entries before and after fill is the *fill overhead*. Our performance model has two basic components: 1) An approximation for the Mflop rate of a matrix with a given block size.

2) An approximation for the amount of unnecessary computation that will be performed due to *fill overhead*.

The first component cannot be exactly determined without running the resulting blocked matrix on each machine of interest. We therefore use an upper bound for this Mflop rate, which is the performance of a dense matrix stored in the blocked sparse format. The second component could be computed exactly for a given matrix, but is quite expensive to compute for multiple block sizes. Instead, we develop an approximation that can be done in a single pass over only a subset of the matrix. These two components differ in the amount of information they require: the first needs the target machine but not the matrix, whereas the second needs the matrix structure but not the machine.

Figure 1 show the performance of sparse matrix vector multiplication for a dense matrix using register-blocked sparse format, on an UltraSPARC I and a MIPS R10000. We vary the block size within a range of values for r and c until the performance degrades. The data in the figure uses a 1000×1000 dense matrix, but the performance is relatively insensitive to the total matrix size as long as the matrix does not fit in cache but does fit in main memory.

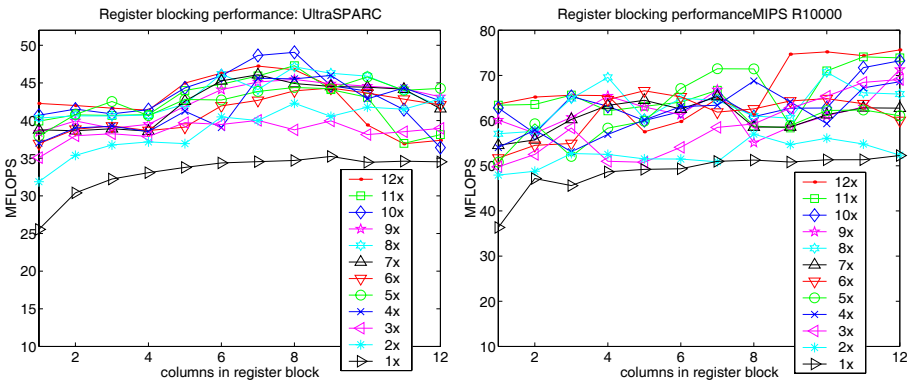


Fig. 1. Performance profile of register-blocked code on an UltraSPARC I (left) and a MIPS R10000 (right): These numbers are taken for a 1000×1000 dense matrix represented in sparse blocked format. Each line is for a fixed number of rows (r), varying the number of columns (c) from 1 to 12.

To approximate the unnecessary computation that would result from register blocking, we estimate the fill overhead. To keep the cost of this computation low, two separate computations are made over the matrix of interest for a column blocking factor (c) and a row blocking factor (r), each being done for a square block size and examining only a fraction of the matrix. For example, to compute r we sample every k^{th} row to compute the fill overhead for that row for every value of r being considered. We use this estimate of fill overhead to predict the performance of an $r \times r$ blocking of a particular matrix A as:

$$\frac{\textit{performance of a dense matrix in } c \times c \textit{ sparse blocked format}}{\textit{estimated } \square ll \textit{ overhead for } c \times c \text{ blocking of } A}$$

While k and the range of r can easily be adjusted, we have found that setting k to 100 and letting r range from 1 to r_{max} is sufficient, where r_{max} is the value of r for which the dense matrix demonstrates its best performance. The value of r is chosen to be the one that maximizes the above performance estimate for $r \times r$ blocks. The choice of c is computed independently by an analogous algorithm on columns. Note that while these two computations use square blocks, the resulting values of r and c may be different.

4 Choosing the Number of Vectors

The question of how many vectors to use when multiplying by a set of vectors is partly dependent on the application and partly on the performance of the multiplication operation. For example, there may be a fixed limit to the number of right-hand sides or convergence of an iterative algorithm may slow as the number of vector increases. If there is a large number of vectors available, and the only concern is performance, the optimization space is still quite complex because there are three parameters to consider: the number of rows and columns in register blocks, and the number of vectors.

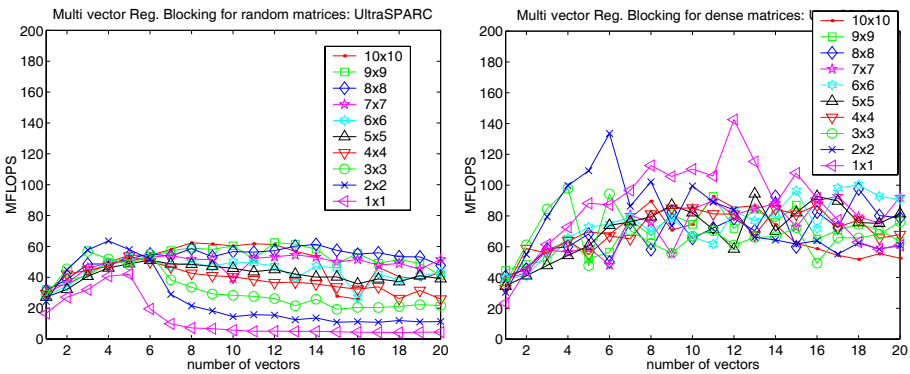


Fig. 2. Register-blocked, multiple vector performance on an UltraSPARC I, varying the number of vectors.

Here we look at the interaction between the register-blocking factors and the number of vectors. This interaction is particularly important because the register-blocked code for multiple vectors unrolls both the register block and multiple vector loops. How effectively the registers are reused in this inner loop is dependent on the compiler. We will simplify the discussion by looking at two

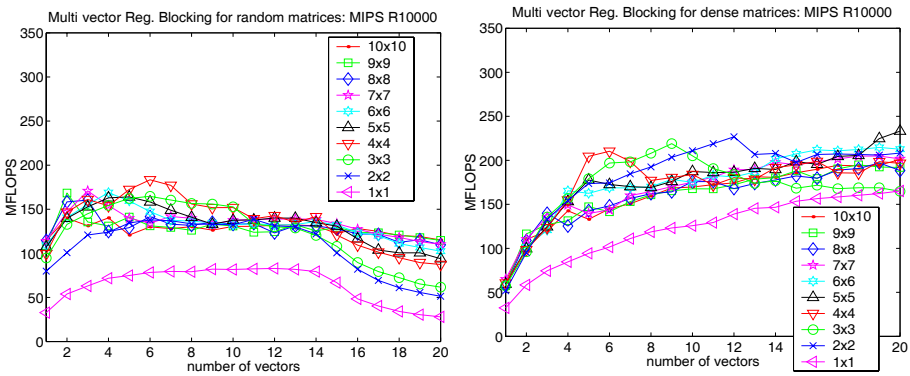


Fig. 3. Register-blocked, multiple vector performance on a MIPS R10000, varying the number of vectors.

extremes in the space of matrix structures: a dense $1K \times 1K$ matrix in sparse format, and sparse $10K \times 10K$ randomly generated matrices with $200K$ (.2%) of the entries being nonzero. In both cases, the matrices are blocked for registers, which in the random cases means that the $200K$ nonzero entries will be clustered differently, depending on the block size. We will also limited our data to square block sizes from 1×1 up to 10×10.

Figures 2 and 3 show the effect of changing the block size and the number of vectors on an UltraSPARC I and MIPS R10000. (The shape of these graphs is different for other machines, but the basic observations below are the same.) The figures shows the performance of register-blocked code optimized for multiple vectors, with the left-hand side showing the randomly structured matrix and the right-hand side showing the dense matrix.

Multiple vectors typically pay off for matrices throughout the regularity and density spectrum, and we can get some sense of this but looking at the dense and random matrices. For most block sizes, even changing from one vector to two is a significant improvement. However, with respect to choosing optimization parameters, the dense and random matrices behave very differently, and there is also quite a bit of variability across machines. There are two characteristics that appear common across both these two machines and others we have studied. First, the random matrix tends to have a peak with some relatively small number of vectors (2-5), whereas for the dense matrix it is at 12 (and generally in the range from 9 to 12 on other machines). For the dense matrix, all of these vectors consume register resources, so the optimal block size is relatively small compared to the that of the single vector code on the same matrix. The behavior of the R10000 is smoother than that of the UltraSPARC, which is probably a reflection of the more expensive memory system on the R10000.

5 Performance of Register Optimizations

We have generated register blocked codes for varying sizes of register blocks and varying numbers of vectors using Sparsity, and have measured their performance on several machines [Im00]. In this paper we will present the results for a set of 39 matrices on the UltraSPARC I and MIPS R10000. The matrices in the set are taken from fluid dynamics, structural modeling, chemistry, economics, circuit simulation and device simulation, and we include one dense matrix in sparse format for comparison. We have omitted matrices from linear programming and information retrieval, which have very little structure and therefore to not benefit from register blocking optimizations. Other optimizations such as cache blocking prove to be useful on some of those.

Figure 5 summarizes the 39 matrices. We have placed the matrices in the table according to our understanding of the application domain from which is was derived. Matrix 1 is a dense matrix. Matrices 2 through 17 are from Finite Element Method (FEM) applications, which in several cases means there are dense sub-locks within much of the matrix. Note however, that the percentage of nonzeros is still very low, so these do not resemble the dense matrix. Matrices 18 through 39 are from structural engineering and device simulation. All the matrices are square, and although some are symmetric, we do not try to take advantage of symmetry here. The matrices are roughly ordered by the regularity of nonzero patterns, with the more regular ones at the top.

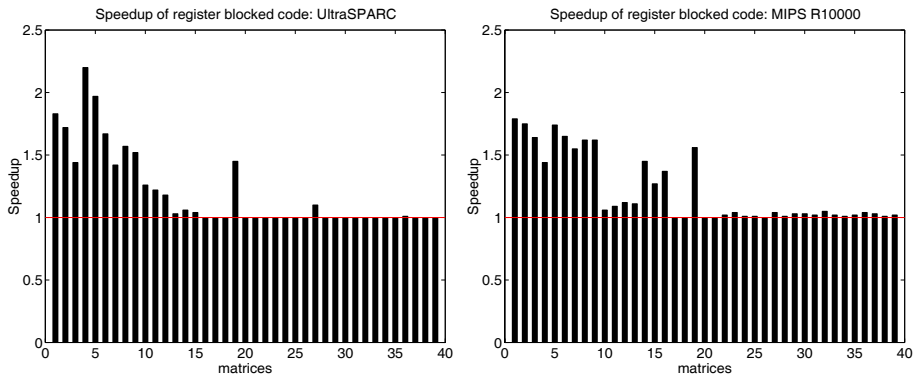


Fig. 4. Speedup of register-blocked multiplication on a 167 MHz UltraSPARC I (left) and a 200MHz MIPS R10000 (right).

Figure 4 shows the effect of register blocking with a single vector on the 39 matrices in table 5. (The Mflop rate was calculated using only those arithmetic operations required by the original representation, not those induced by fill from blocking.) The benefit is highest for the lower numbered matrices, which tend to have naturally occurring dense subblocks, although they are not uniform, so

	Name	Application Area	Dimension	Nonzeros	Sparsity
1	dense1000	Dense Matrix	1000x 1000	1000000	100
2	raefsky3	Fluid structure interaction	21200x21200	1488768	0.33
3	inaccura	Accuracy problem	16146x16146	1015156	0.39
4	bcsstk35	Stiff matrix automobile frame	30237x30237	1450163	0.16
5	venkat01	Flow simulation	62424x62424	1717792	0.04
6	crystk02	FEM Crystal free vibration	13965x13965	968583	0.50
7	crystk03	FEM Crystal free vibration	24696x24696	1751178	0.29
8	nasasrb	Shuttle rocket booster	54870x54870	2677324	0.09
9	3dtube	3-D pressure tube	45330x45330	3213332	0.16
10	ct20stif	CT20 Engine block	52329x52329	2698463	0.10
11	bai	Airfoil eigenvalue calculation	23560x23560	484256	0.09
12	raefsky4	buckling problem	19779x19779	1328611	0.34
13	ex11	3D steady flow calculation	16614x16614	1096948	0.40
14	rdist1	Chemical process separation	4134x 4134	94408	0.55
15	vavasis3	2D PDE problem	41092x41092	1683902	0.10
16	orani678	Economic modeling	2529x 2529	90185	1.41
17	rim	FEM fluid mechanics problem	22560x22560	1014951	0.20
18	memplus	Circuit Simulation	17758x17758	126150	0.04
19	gemat11	Power flow	4929x 4929	33185	0.14
20	lhr10	Light hydrocarbon recovery	10672x10672	232633	0.20
21	goodwin	Fluid mechanics problem	7320x 7320	324784	0.61
22	bayer02	Chemical process simulation	13935x13935	63679	0.03
23	bayer10	Chemical process simulation	13436x13436	94926	0.05
24	coater2	Simulation of coating flows	9540x 9540	207308	0.23
25	finan512	Financial portfolio optimization	74752x74752	596992	0.01
26	onetone2	Harmonic balance method	36057x36057	227628	0.02
27	pwt	Structural engineering problem	36519x36519	326107	0.02
28	vibrobox	Structure of vibroacoustic problem	12328x12328	342828	0.23
29	wang4	Semiconductor device simulation	26068x26068	177196	0.03
30	lnsp3937	Fluid flow modeling	3937x 3937	25407	0.16
31	lns3937	Fluid flow modeling	3937x 3937	25407	0.16
32	sherman5	Oil reservoir modeling	3312x 3312	20793	0.19
33	sherman3	Oil reservoir modeling	5005x 5005	20033	0.08
34	orsreg1	Oil reservoir simulation	2205x 2205	14133	0.29
35	saylr4	Oil reservoir modeling	3564x 3564	22316	0.18
36	shyy161	Viscous flow calculation	76480x76480	329762	0.01
37	wang3	Semiconductor device simulation	26064x26064	177168	0.03
38	mcfe	astrophysics	765x 765	24382	4.17
39	jpwh991	Circuit physics modeling	991x 991	6027	0.61

Fig. 5. Matrix benchmark suite: The basic characteristic of each matrix used in our experiments is shown. The sparsity column is the percentage of nonzeros.

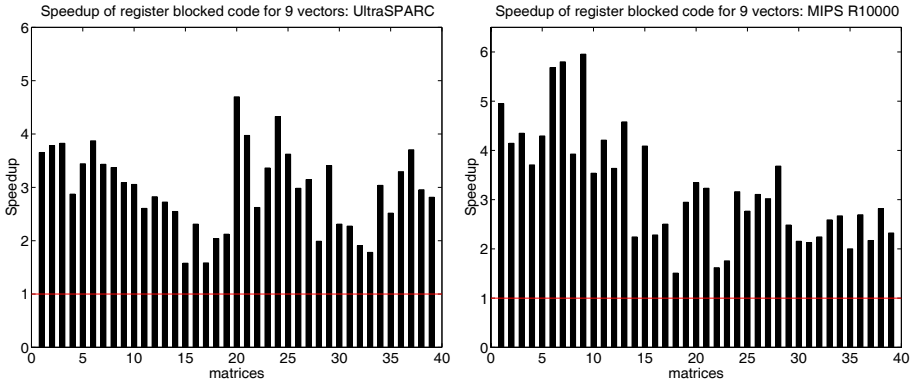


Fig. 6. Speedup of register-blocked, multiple vector code using 9 vectors.

there is fill overhead. Some of the matrices that have no natural subblocks still benefit from small blocks.

Figure 6 shows the speedup of register blocking for multiple vectors on a same matrix set. The number of vectors is fixed at 9, and it shows a tremendous payoff. On the MIPS R10000, the lower-number matrices have a slight advantage, and on the UltraSPARC, the middle group of matrices sees the highest benefit; these are mostly matrices from scientific simulation problems with some regular patterns, but without the dense sub-blocks that appear naturally in the lower-numbered FEM matrices. Overall, benefits are much more uniform across matrices than for simple register blocking.

6 Related Work

Sparsity is related to several other projects that automatically tune the performance of algorithmic kernels for specific machines. In the area of sparse matrices, these systems include the *sparse compiler* that takes a dense matrix program as input and generates code for a sparse implementation [Bik96]. As in Sparsity, the matrix is examined during optimization, although the sparse compiler looks for higher level structure, such as bands or symmetry. This type of analysis is orthogonal to ours, and it is likely that the combination would prove useful. The Bernoulli compiler also takes a program written for dense matrices and compiles it for sparse ones, although it does not specialize the code to a particular matrix structure. Toledo [Tol97] demonstrated some of the performance benefits or register blocking, including a scheme that mixed multiple block sizes in a single matrix, and PETSc (Portable, Extensible Toolkit for Scientific Computation) [BGMS00] uses a application-specified notion of register blocking for Finite Element Methods. Toledo and many others have explored the benefits of reordering sparse matrices, usually for parallel machines or when the natural ordering of the application has been destroyed. Finally, we note that the BLAS Technical

Forum has already identified the need for runtime optimization of sparse matrix routines, since they include a parameter in the matrix creation routine to indicate how frequently matrix-vector multiplication will be performed [BLA99].

7 Conclusions

In this paper, we have described optimization techniques to increase register reuse in sparse matrix-vector multiplication for one or more vectors. We described some parts of the Sparsity system that generate code for fixed block sizes, filling in zeros as necessary. To select the register block size, we showed that a simple performance model that separately takes a machine performance profile and a matrix fill estimation worked very well. The model usually chooses the optimal block size, producing speedups of around 2× for some matrices. Even on matrices where the blocks were not evident at the application level, small blocks proved useful on some machines. We also extended the Sparsity framework to generate code for multiple vectors, where the benefits are are high as 5× on the machines and matrices shown here.[1]

References

[BCD+00] Z. Bai, T.-Z. Chen, D. Day, J. Dongarra, A. Edelman, T. Ericsson, R. Freund, M. Gu, B. Kagstrom, A. Knyazev, T. Kowalski, R. Lehoucq, R.-C. Li, R. Lippert, K. Maschoff, K. Meerbergen, R. Morgan, A. Ruhe, Y. Saad, G. Sleijpen, D. Sorensen, and H. Van der Vorst. Templates for the solution of algebraic eigenvalue problems: A practical guide. in preparation, 2000.

[BGMS00] Satish Balay, William D. Gropp, Lois Curfman McInnes, and Barry F. Smith. PETSc 2.0 users manual. Technical Report ANL-95/11 - Revision 2.0.28, Argonne National Laboratory, 2000.

[Bik96] Aart J. C. Bik. *Compiler Support for Sparse Matrix Computations.* PhD thesis, Leiden University, 1996.

[BLA99] BLAST Forum. *Documentation for the Basic Linear Algebra Subprograms (BLAS)*, October 1999. http://www.netlib.org/blast/blast-forum.

[Im00] Eun-Jin Im. *Optimizing the Performance of Sparse Matrix - Vector Multiplication.* PhD thesis, University of California at Berkeley, May 2000.

[LRW91] M. S. Lam, E. E. Rothberg, and M. E. Wolf. The cache performance and optimizations of blocked algorithms. In *Proceedings of the Fourth International Conference on Architectural Support for Programming Languages and Operating Systems*, April 1991.

[Mar95] Osni A. Marques. BLZPACK: Decsription and User's guide. Technical Report TR/PA/95/30, CERFACS, 1995.

[Tol97] Sivan Toledo. Improving memory-system performance of sparse matrix-vector multiplication. In *Proceedings of the 8th SIAM Conference on Parallel Processing for Scientific Computing*, March 1997.

[1] This research is supported in part by U.S. Army Research Office, by the Department of Energy and by Kookmin University, Korea.

Rescheduling for Locality in Sparse Matrix Computations

Michelle Mills Strout, Larry Carter, and Jeanne Ferrante

University of California, San Diego

Abstract. In modern computer architecture the use of memory hierarchies causes a program's data locality to directly affect performance. Data locality occurs when a piece of data is still in a cache upon reuse. For dense matrix computations, loop transformations can be used to improve data locality. However, sparse matrix computations have non-affine loop bounds and indirect memory references which prohibit the use of compile time loop transformations. This paper describes an algorithm to tile at runtime called *serial sparse tiling*. We test a runtime tiled version of sparse Gauss-Seidel on 4 different architectures where it exhibits speedups of up to 2.7. The paper also gives a static model for determining tile size and outlines how overhead affects the overall speedup.

1 Introduction

In modern computer architecture the use of memory hierarchies causes a program's data locality to directly affect performance. Data locality occurs when a piece of data is still in the cache upon reuse. This paper presents a technique for tiling sparse matrix computations in order to improve the data locality in scientific applications such as Finite Element Analysis.

The Finite Element Method (FEM) is a numerical technique used in scientific applications such as Stress Analysis, Heat Transfer, and Fluid Flow. In FEM the physical domain being modeled is discretized into an unstructured grid or mesh (see figure 3). FEM then generates simultaneous linear equations that describe the relationship between the unknowns at each node in the mesh. Typical unknowns include temperature, pressure, and xy-displacement. These equations are represented with a sparse matrix A and vectors $\boldsymbol{u}$ and $\boldsymbol{f}$ such that $A\boldsymbol{u} = \boldsymbol{f}$.

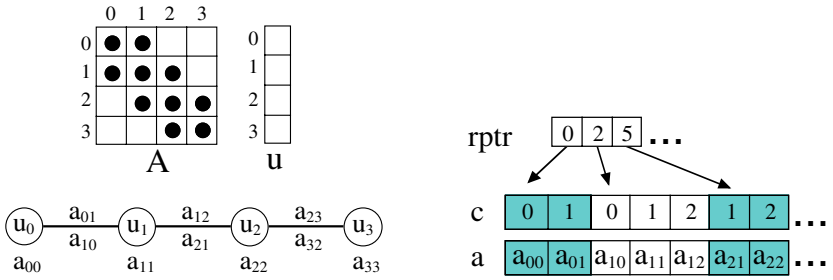
Conjugate Gradient, Gauss-Seidel and Jacobi are all iterative methods for solving simultaneous linear equations. They solve for $\boldsymbol{u}$ by iterating over the sparse matrix A a constant number of times, converging towards a solution. The iteratively calculated value of a mesh node unknown u_j depends on the values of other unknowns on the same node, the unknowns associated with adjacent nodes within the mesh, and the non-zeros/coefficients in the sparse matrix which relate those unknowns. Typically the sparse matrix is so large that none of the values used by one calculation of u_j remain in the cache for future iterations on u_j, thus the computation exhibits poor data locality.

For dense matrix computations, compile time loop transformations such as tiling or blocking [17] can be used to improve data locality. However, since sparse

V.N. Alexandrov et al. (Eds.): ICCS 2001, LNCS 2073, pp. 137–146, 2001.
© Springer-Verlag Berlin Heidelberg 2001

matrix computations operate on compressed forms of the matrix in order to avoid storing zeros, the loop bounds are not affine and the array references include indirect memory references such as $a[c[i]]$. Therefore, straightforward application of tiling is not possible. In this paper, we show how to extend tiling via runtime reorganization of data and rescheduling of computation to take advantage of the data locality in such sparse matrix computations.

Specifically, we reschedule the sparse Gauss-Seidel computation at runtime. First we tile the iteration space and then generate a new schedule and node numbering which allows each tile to be executed atomically. Typically the numbering of the nodes in the mesh is arbitrary, therefore, renumbering the nodes and maintaining the Gauss-Seidel partial order on the new numbering allows us to still use the convergence theorems for Gauss-Seidel. The goal is to select the tile size so that the tile only touches a data subset, which fits into cache.



(a) Logical data associations (b) Actual CSR storage format

Fig. 1. Data associated with Mesh

To illustrate, we look at an example of how one would tile the Gauss-Seidel computation on a one-dimensional mesh. Figure 1(a) shows how we can visualize what data is associated with each node in the mesh. The unknown values being iteratively updated are associated with the nodes, [1] and the coefficients representing how the unknowns relate are associated with the edges and nodes. However, keep in mind that the matrix is stored in a compressed format like compressed sparse row (see figure 1(b)) to avoid storing the zeros.

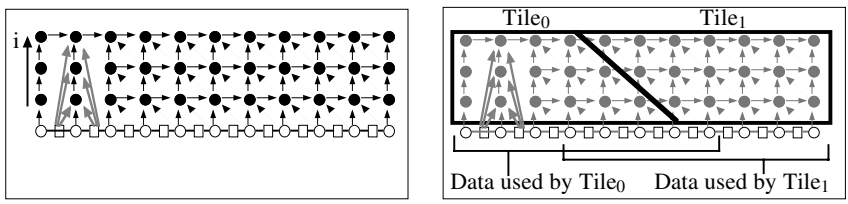
The pseudo-code for Gauss-Seidel is shown below. The outermost loop iterates over the entire sparse matrix generated by solving functions on the mesh. We refer to the i iterator as the *convergence iterator*. The j loop iterates over the rows in the sparse matrix. [2] The k loop which is implicit in the summations iterates over the unknowns which are related to u_j, with $a_{jk}u_k^{(i)}$ and $a_{jk}u_k^{(i-1)}$ only being computed when a_{jk} is a non-zero value.

[1] In this example there is only one unknown per mesh node.

[2] There is one row in the matrix for each unknown at each mesh node.

for $i = 1, 2, ..., T$
  for $j = 1, 2, ..., R$
    $u_j^{(i)} = (1/a_{jj})(f_j - \sum_{k=1}^{j-1} a_{jk} u_k^{(i)} - \sum_{k=j+1}^{n} a_{jk} u_k^{(i-1)})$

The Gauss-Seidel computation can be visualized with the iteration space graph shown in figure 2. Each black iteration point [3] , $< i, v >$, represents the computations for all $u_j^{(i)}$ where u_j is an unknown associated with mesh node v and i is the convergence iteration. The initial values associated with a 1D mesh are shown in white. The arrows represent data dependences [4] that specify when an initial value or a value generated by various iteration points is used by other iteration points. We refer to each set of computation for a particular value of i within the iteration space as a *layer*. Figure 2 contains three layers of computation over a mesh.



(a) Original Computation (b) Divided into 2 Tiles

Fig. 2. Gauss-Seidel Iteration Space Graph

Notice that the sparse matrix values associated with the edges adjacent to a particular mesh node v are reused in each computation layer. However, the mesh is typically so large that upon reuse the matrix entries are no longer in the cache. To improve the computation's data locality, we reschedule it based on a tiling like the one shown in figure 2(b). The resulting schedule executes all of the iteration points in one tile before continuing on to the next tile; in other words, each tile is executed atomically. By choosing an appropriate tile size the data used by each tile will fit into cache for all instances of $< i, v >$ within the tile and therefore improve the data locality of the computation.

In Section 2 we present the algorithm which tiles and reschedules Gauss-Seidel at runtime. Then in section 3 we give experimental results which show that improving the data locality does improve code performance. We also outline the affect of overhead and how to select tile sizes. Finally, we present some related work and conclusions.

[3] We use the term iteration point for points in the iteration space graph and node for points in the mesh.
[4] Some dependences are omitted for clarity.

2 Tiling Sparse Computations

In order to tile the iteration space induced by the convergence iteration over the mesh, we partition the mesh and then grow tiles backwards through the iteration space based on the seed partitions. Figure 3 shows the iteration space for a 2D mesh with each layer drawn separately. Edges show the connectivity of the underlying mesh. We use the resulting tiling to reschedule the computation and renumber the nodes in the mesh. Since tiles depend on results calculated by neighboring tiles, the tiles must be executed in a partial order which respects those dependences.

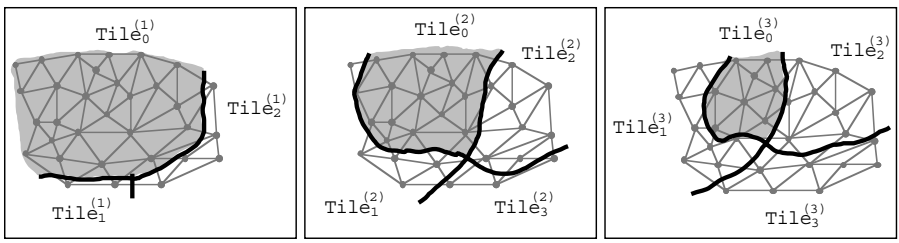


Fig. 3. Tile layers for $Tile_0$, $Tile_1$, $Tile_2$, and $Tile_3$. The tile layers for $Tile_0$ are shaded.

We refer to the runtime tiling of sparse matrix computations as *sparse tiling*. This paper describes and implements a *serial sparse tiling*, in that the resulting schedule is serial. Douglas et al. [4] describe a *parallel sparse tiling* for Gauss-Seidel. They partition the mesh and then grow tiles forward through the iteration space (in the direction of the convergence iterator) in such a way that the tiles do not depend on one another and therefore can be executed in parallel. After executing the tiles resulting from parallel sparse tiling, it is necessary to execute a fill-in stage which finishes all the iteration points not included in the tiles. Future work includes determining when to use a serial sparse tiling or a parallel sparse tiling based on the target architecture and problem size.

Both sparse tiling strategies follow the same overall process at runtime.

1. **Partition** the mesh
2. **Tile** the iteration space induced by the partitioned mesh
3. **Reschedule** the computation
4. **Execute** the new schedule

The next sub-sections describe each part of the process for the serial sparse tiling strategy which we have developed.

2.1 Partition

Although graph partitioning is an NP-Hard problem [6], there are many heuristics used to get reasonable graph partitions. We use the Metis [11] software package to do the partitioning at runtime on the mesh. The partitioning algorithm in Metis has a complexity of $O(|E|)$ where $|E|$ is the number of edges in the mesh [12].

2.2 Tiling

Recall the iteration space for sparse Gauss-Seidel shown in figure 2 where each iteration point represents values being generated for the unknowns on the associated mesh node v at convergence iteration i. A tile within this space is a set of layers, one per each instance of the convergence iterator i. Each tile layer computes the values for a subset of mesh nodes. The final layer of a tile (see the last layer in figure 3) corresponds to the nodes in one partition, p, of the mesh. The tile layers for earlier convergence iterations are formed by adding or deleting iteration points from the seed partition to allow atomic execution of the tile without violating any data dependences.

To describe the sparse serial tiling algorithm for sparse Gauss-Seidel we use the following terminology. The mesh can be represented by a graph $G(V, E)$ consisting of a set of nodes V and edges E. An **iteration point**, $< i, v >$, represents the computation necessary at convergence iteration i for the unknowns associated with node v. A **tile**, $Tile_p$, is a set of iteration points that can be executed atomically. Each tile is designated by an integer identifier p, which also represents the execution order of the tiles. A **tile layer**, $Tile_p^{(i)}$, includes all iteration points within tile p being executed at convergence iteration i.

The tiling algorithm generates a function θ that returns the identifier for the tile which is responsible for executing the given iteration point, $\theta(< i, v >) : I$ x $V \rightarrow \{0, 1, ..., m\}$, where m is the number of tiles. $Tile_0$ will execute all vertex iterations with $\theta(< i, v >) = 0$, $Tile_1$ will execute all vertex iterations with $\theta(< i, v >) = 1$, etc. A **tile vector**, $\Theta(j) =< \theta(< 1, v >), ..., \theta(< T, v >) >$, stores tile identifiers for all the tiles which will be executing iteration points for a specific node in the mesh.

The algorithm shown below gives all nodes a legal tile vector. It takes as input the *part* function, $part(v) : V \rightarrow \{1, 2, ...m\}$, which is the result of the mesh partitioning. The *part* function specifies a partition identifier for each mesh node. Recall that we will be growing one tile for each seed partition. The first step in the algorithm is to initialize all tile vectors so that each iteration point is being executed by the tile being grown from the associated mesh node's partition in the mesh. $Worklist(T)$ is then initialized with all nodes. The loop then grows the tiles backward from $i = T$ by adding and removing iteration points as needed in order to maintain the data dependences. A detailed explanation of this loop is omitted due to space constraints.

Algorithm AssignTileVector(part)

(1) $\forall v \in V, \Theta(v) =< part(v), part(v), ..., part(v) >$
(2) $Worklist(T) = V$
(3) for $i = T$ downto 2
(4) for each node $v \in Worklist(i)$
(5) for each $(v, w) \in E$
(6) if $w \not\in Worklist(i-1)$ then
(7) if $\theta(< i-1, w >) > \theta(< i, v >)$ then
(8) $w \in Worklist(i-1)$
(9) $\forall q$ st. $1 \leq q \leq (i-1)$, $\theta(< q, w >) \longleftarrow \theta(< i, v >)$

An upper bound on the complexity of this algorithm is $O(T|E|)$ or equivalently $O(\frac{TZ}{d^2})$ where d is the degrees of freedom, $|E|$ is the number of edges in the mesh, Z is the number of non-zeros in the sparse matrix, and T is the number of convergence iterations the Gauss-Seidel algorithm will perform.

2.3 Renumbering and Rescheduling

The mesh nodes are renumbered in lexicographical order of their corresponding tile vectors. The lexicographical order insures that the resulting schedule will satisfy the Gauss-Seidel partial order on the new numbering. We schedule all the computations in $Tile_p$ before any in $Tile_{p+1}$, and within a tile we schedule the computations by layer and within a layer.

2.4 Execute Transformed Computation

Finally, we rewrite the sparse Gauss-Seidel computation to execute the new schedule. The new schedule indicates which iteration points should be executed for each tile at each convergence iteration.

3 Experimental Results for Gauss-Seidel

To evaluate the possible benefits of our approach, we compare the performance of the Gauss-Seidel routine in the finite element package FEtk [9] with a runtime tiled and rescheduled version of the same algorithm. For input, we use the sparse matrices generated for a nonlinear elasticity problem on 2D and 3D bar meshes. We generate different problem sizes by using FEtk's adaptive refinement. The rescheduled code runs on an Intel Pentium III, an IBM Power3 node on the Blue Horizon at the San Diego Supercomputer Center, a Sun UltraSparc-IIi, and a DEC Alpha 21164.

When not taking overhead into account the new schedule exhibits speedups between 0.76 (a slowdown) and 2.7 on the four machines, see figure 4. Next we describe the simple static model used for selecting the partition size - the main tuning parameter for the new schedule. Finally we outline the effect overhead will have on the overall speedup.

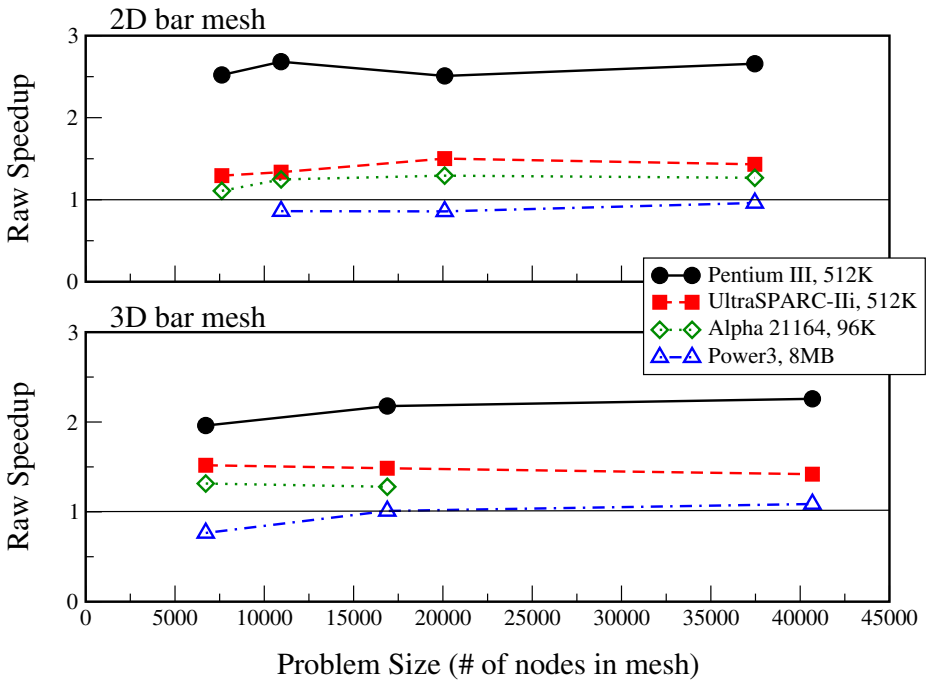


Fig. 4. Speedups over FEtk's Gauss-Seidel for 2D and 3D bar mesh without adding overhead. The partition size was selected to fit into the L2 cache on each machine whose sizes are shown in the legend.

3.1 Partition Size Selection

Before tiling and rescheduling at runtime the available parameters are the number of nodes in the mesh, the number of unknowns per vertex, the number of convergence iterations, and the cache size of the target architecture. Using this information we want to determine which partition size will generate tiles which fit into a level of cache and therefore improve performance.

In Gauss-Seidel for each unknown at each mesh node we iteratively compute $w_j = f_j - \sum_{k>i} a_{jk} * u_j$ and $u_j = (w_j - \sum_{k<j} a_{jk} * u_j)/a_{jj}$. Using K as the average number of neighbors each node has in the mesh and d as the number of unknowns per mesh node, the computation will use $3 * d$ scalars for the vectors u, w, and f and $K * d^2$ associated non-zeros from the sparse matrix A while updating the unknowns for each mesh node. If we assume compressed sparse row (CSR) storage format then the amount of memory needed by N mesh nodes is $Mem(N) = N * K * d^2 * sizeof(double) + N * K * d^2 * sizeof(int) + 3 * N * d * sizeof(double)$. In all of the experiments we solve for N such that only half the memory in the L2 cache of the machine is utilized.

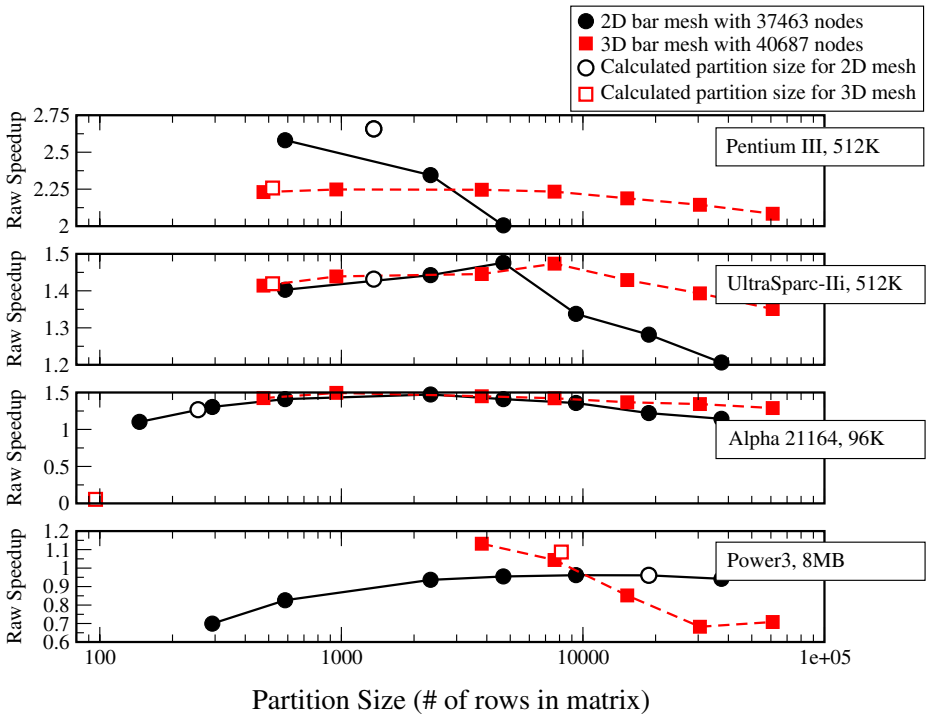


Fig. 5. How the partitions size affects speedup for 5 convergence iterations on 4 different machines. The outlined symbols represent partitions sizes calculated by the model.

Figure 4 shows the speedups on different mesh sizes when the partition sizes are selected in this manner. When compared to the speedups over a variety of partitions sizes the calculated partition sizes do reasonably well, see figure 5, except on the Alpha. This is probably due to the small L2 size on the Alpha and the existence of a 2MB L3 cache.

3.2 Overhead

Figures 4 and 5 show the speedups obtained by the tiled and rescheduled code over FEtk's implementation of Gauss-Seidel without taking overhead into account. It is important to look at the speedups without overhead because Gauss-Seidel can be called multiple times on the same mesh within an algorithm like Multigrid. Therefore, even though overhead might make rescheduling not beneficial for one execution of the Gauss-Seidel computation, when amortized over multiple calls to the computation we get an overall speedup.

By looking at the calculated partition sizes resulting in the highest (2.68) and lowest [5] (1.11) speedups we see that the tiled and rescheduled version of

[5] Ignoring the Power3 results because speedup was less than 1.

Gauss-Seidel would need to be called between 5 to 27 times in order to observe an overall speedup. However, there were partition sizes not calculated by our simple static model which resulted in an overall speedup with only 1 call to the tiled and rescheduled Gauss-Seidel. For example, on a 3D mesh with $N = 40,687$ an overall speedup of 1.17 is observed even when the overhead cost is included. This indicates that the tradeoff between raw speedup and overhead must be considered when calculating partition sizes.

4 Related Work

Douglas et al. [4] does tiling on the iteration space graph resulting from unstructured grids in the context of the Multigrid algorithm using Gauss-Seidel as a smoother. They achieve overall speedups up to 2 with 2D meshes containing 3983, 15679, and 62207 nodes on an SGI O2. They are able to reschedule their tiles in parallel and then finish the remaining computation with a serial backtracking step. Our technique also tiles the Gauss-Seidel iteration space, but we execute our tiles serially in order to satisfy dependences between tiles. Also, we do not require a backtracking step which exhibits poor data locality. These two tiling algorithms are instances of a general class of temporal locality transformations which we will refer to as sparse tiling.

Mitchell et al [14] describe a compiler optimization which operates on non-affine array references in code. The use of sparse data structures causes indirect array references which are a type of non-affine array reference. Also, Eun-Jin Im [10] describes a code generator called SPARSITY which generates cache-blocked sparse matrix-vector multiply. Both of these techniques improve spatial and temporal locality on the vectors $\boldsymbol{u}$ and $\boldsymbol{f}$ when dealing with the system $A\boldsymbol{u} = \boldsymbol{f}$. However, they do not improve the temporal locality on the sparse matrix, because in their rescheduled code the entire sparse matrix is traversed each convergence iteration. Other work which looks at runtime data reorganization and rescheduling includes Demmel et al. [2], Han and Tseng[8], Ding and Kennedy [3], and Mellor-Crummey et al.[13].

5 Conclusion

Runtime tiling is possible with unstructured iteration spaces, and we show it can improve the data locality and therefore the performance of Gauss-Seidel. Specifically we present an algorithm for generating a serial sparse tiling for Gauss-Seidel. We also describe a simple static model for selecting partition sizes from which the tiles are grown. Future work includes improving the model used to calculate partition sizes so that the tradeoff between overhead and raw speedup is taken into account. Also, the performance model needs to determine when to use a serial sparse tiling or a parallel sparse tiling based on the target architecture and problem size.

References

1. Jeff Bilmes, Krste Asanović, Chee whye Chin, and Jim Demmel. Optimizing matrix multiply using PHiPAC: a Portable, High-Performance, ANSI C coding methodology. In *Proceedings of International Conference on Supercomputing*, Vienna, Austria, July 1997.
2. James W. Demmel, Stanley C. Eisenstat, John R. Gilbert, Xiaoye S. Li, and Joseph W. H. Liu. A supernodal approach to sparse partial pivoting. *SIAM Journal on Matrix Analysis and Applications*, 20(3):720–755, July 1999.
3. Chen Ding and Ken Kennedy. Improving cache performance in dynamic applications through data and computation reorganization at run time. In *Proceedings of the ACM SIGPLAN '99 Conference on Programming Language Design and Implementation*, pages 229–241, Atlanta, Georgia, May 1–4, 1999.
4. Craig C. Douglas, Jonathan Hu, Markus Kowarschik, Ulrich Rüde, and Christian Weiss. Cache Optimization for Structured and Unstructured Grid Multigrid. *Electronic Transaction on Numerical Analysis*, pages 21–40, February 2000.
5. Matteo Frigo and Steven G. Johnson. Fftw: An adaptive software architecture for the fft. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, page 1381, 1998.
6. Michael R. Garey, David S. Johnson, and L. Stockmeyer. Some simplified *NP*-complete graph problems. *Theoretical Computer Science*, 1:237–267, 1976.
7. Kang Su Gatlin. Portable High Performance Programming *via Architecture Cognizant Divide-and-Conquer Algorithms*. Ph.d. thesis, University of California, San Diego, September 2000.
8. Hwansoo Han and Chau-Wen Tseng. Efficient compiler and run-time support for parallel irregular reductions. *Parallel Computing*, 26(13–14):1861–1887, December 2000.
9. Michael Holst. Fetk = the finite element toolkit. http://www.fetk.org.
10. Eun-Jin Im. *Optimizing the Performance of Sparse Matrix-Vector Multiply*. Ph.d. thesis, University of California, Berkeley, May 2000.
11. George Karypis and Vipin Kumar. *Metis: A Software Package for Partitioning Unstructured Graphs, Partitioning Meshes, and computing Fill-Reducing Orderings of Sparse Matrices Version 4.0*, 1998.
12. George Karypis and Vipin Kumar. Multilevel k-way partitioning scheme for irregular graphs. *Journal of Parallel and Distributed Computing*, 48(1):96–129, 10 January 1998.
13. John Mellor-Crummey, David Whalley, and Ken Kennedy. Improving memory hierarchy performance for irregular applications. In *Proceedings of the 1999 Conference on Supercomputing*, ACM SIGARCH, pages 425–433, N.Y., June 20–25 1999. ACM Press.
14. Nicholas Mitchell, Larry Carter, and Jeanne Ferrante. Localizing non-affine array references. In *Proceedings of the 1999 International Conference on Parallel Architectures and Compilation Techniques (PACT '99)*, pages 192–202, Newport Beach, California, October 12–16, 1999. IEEE Computer Society Press.
15. Nick Mitchell. Guiding Program Transformations *with Modal Performance Model. Ph.d. thesis, University of California, San Diego, August 2000.*
16. R. Clint Whaley and Jack J. Dongarra. Automatically tuned linear algebra software. In *Supercomputer 98*, 1998.
17. Michael J. Wolfe. *High Performance Compilers for Parallel Computing*. Addison-Wesley, 1996.

The DOE Parallel Climate Model (PCM): The Computational Highway and Backroads

Thomas Bettge, Anthony Craig, Rodney James,
Vincent Wayland, and Gary Strand

National Center for Atmospheric Research, 1850 Table Mesa Drive,
Boulder, Colorado 80303
(bettge, tcraig, rodney, wayland, strandwg) @ucar.edu

Abstract. The DOE Parallel Climate Model (PCM) is used to simulate the earth's climate system and has been used to study the climate of the 20th century and to project possible climate changes into the 21st century and beyond. It was designed for use on distributed memory, highly parallel, architectures. The computational requirements and design of the model are discussed, as well as its performance and scalability characteristics. A method for port validation is demonstrated. The shortcomings of the current model are summarized and future design plans are presented.

1 Introduction

The U.S. Department of Energy (DOE) Parallel Climate Model (PCM), constructed at the National Center for Atmospheric Research (NCAR) during the mid-1990s, is a comprehensive model of the earth's climate system, including fully active, scientifically contemporary, components of the atmosphere, ocean, land, river runoff, and sea ice.

The model has been used to simulate numerous climate states and climate change scenarios, including control climates of 1870, 1990, and 2060 (before and after the onset of anthropogenic activities), and the transient climate during 1870-2000 using observed greenhouse gas, sulfate aerosol, and solar variance forcing (Washington et al., 2000). Ensembles of future simulations extending to 2100 and beyond have been performed using the Intergovernmental Panel On Climate Change (IPCC) Third Assessment Report best case and worse case greenhouse gas forcing scenarios (Cubasch et al., 2001). In all, the PCM has been used to simulate over 7000 years of the earth's climate during the past four years.

At inception the target architectures of the PCM were massively parallel, distributed memory machines. The mid-1990s generation of Massively Parallel Processing (MPP) computer architectures forced the design of PCM into an all message passing application where the components themselves were stacked into the local memory on each computational node and run sequentially. Subsequently, as computer hardware

V.N. Alexandrov et al. (Eds.): ICCS 2001, LNCS 2073, pp. 149-158, 2001.
© Springer-Verlag Berlin Heidelberg 2001

at DOE facilities evolved with the deployment of Distributed Shared Memory (DSM) architectures and clusters of Symmetric Multi-Processors (SMPs), the PCM was ported and used successfully on a variety of these types of machines.

While the primary focus of the PCM is scientific investigation and discovery, computational performance is a high priority. The purpose of this paper is to review and present several computational aspects of the PCM including the early decision making process, the issue of portability across machines, and performance and productivity on available machines.

2 Computational Requirements

A major goal of the PCM was to use existing model components developed under the DOE Computer Hardware Advanced Mathematics Model Physics (CHAMMP) program. These individual models were targeted for machines with distributed memory across a large number of processors – the MPP class of architectures.

Many of the computational requirements of the PCM are obvious and include:

- must easily port across a number of platforms
- must validate across a number of platforms (i.e., must produce statistically identical climate simulations)
- must achieve a minimum level of high performance
- must use existing component models (PCM was *not* a model/software development project, but rather was charged to modify existing models to the extent necessary to achieve the requirements in a substantial way)

With respect to software development, the PCM was created using software engineering techniques which could focus on rapid development with customized software specific to, and documented among, a small group of developers. The limited scope of the project did not require the overhead and management which is recommended for larger projects which develop software for a much larger community.

3 The Model Components

The PCM is composed of individual models of the atmosphere, land, ocean, river runoff, and sea ice.

The atmospheric model is the parallel version of the NCAR Community Climate Model version 3 (CCM3) at T42 spectral truncation horizontal resolution (2.8 degrees latitude by longitude) with 18 vertical levels (see Kiehl et al., 1998).

The land surface model is a one-dimensional model of the energy, momentum, and water exchanges between the earth's surface and the atmosphere, on the same horizontal grid as the atmosphere (Bonan, 1996). A river runoff scheme is included which directs freshwater from the land into the ocean according to predetermined (observed) basin flows (Branstetter et al., 2001).

The ocean component is the Los Alamos National Laboratory Parallel Ocean Program (POP) model with a displaced polar grid in the northern hemisphere. The grid has an average horizontal resolution of 2/3 degress latitude and longitude, with 32 vertical levels (Dukowicz and Smith, 1994).

The sea-ice model provides a full dynamic and thermodynamic representation of the polar ice caps on a 27 km Cartesian grid centered over each pole (Washington et al., 2000).

Since the PCM is composed of components from independent model developers, considerable effort was required to create a software environment where the full PCM system would perform optimally, including the construction and implementation of a so-called *coupler*, which facilitates the exchange of information between individual components. The coupler steers the execution of the components and provides the mechanisms necessary to coordinate and support the transfer of data between the components, including computation of fluxes at the component interfaces, conservation of energy, and interpolation of data from one grid to another using the technique of Jones (1999).

4 Computational Implementation

When considering the implementation of the various components of PCM into a fully integrated computational model, it was necessary to evaluate the suitability of the available hardware to the problem at hand. Since the original target, the Cray-T3D/T3E, did not have at the time an acceptable mechanism for allowing high speed message passing between active processes on separate partitions, it was decided to implement the PCM as a Single Program Multiple Data (SPMD) application, overlaying each component within the memory on the local compute node of the partition.

Fig. 1 shows a simplified representation of how the PCM is integrated sequentially in time. The primary time period over which the PCM components are synchronized, defined as the *synchronous coupling interval*, is one day. During this period the components each advance one day in sequence, with each component gathering or passing data as needed to the coupler for processing and distribution to the next component. All components and the coupler use *all* of the available computational nodes during the period in which they are active. Message passing within each component is achieved by use of the standard Message Passing Interface (MPI) library.

Two major obstacles encountered in this scheme are (1) the horizontal grid is different for each component, and (2) the internal data decomposition employed in the distributed partitioning is different for each component. Thus, the partitioning strategy in the coupler became one of optimizing the use of MPI to gather and scatter information as needed when it is processed (in interpolating data from one grid to another, for example). Since the exchange of data between components involves very few floating point operations, the coupler activity is inherently non-scalable. For the PCM, this data movement problem was minimized by passing a majority of the data between components at the day boundaries.

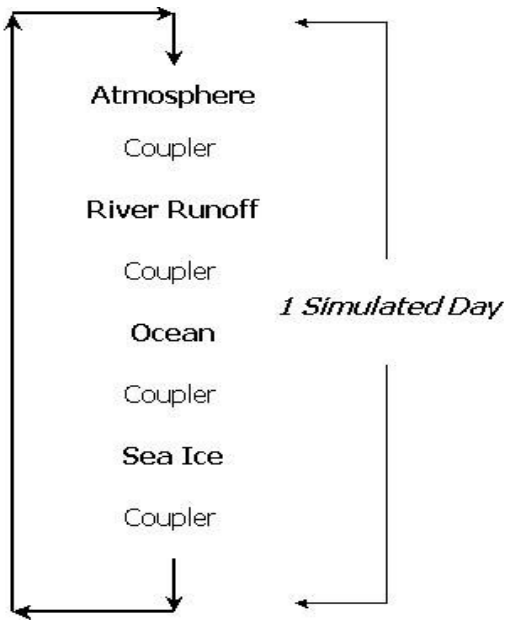


Fig. 1. Simplified integration sequence of the component models in PCM over the synchronous time interval of one day.

There are two characteristics of the current components used in PCM which limit its overall performance and scalability. First, the atmospheric model contains a one-dimensional decomposition, which limits the number of processors that can be used in the solution to the number of latitude lines of the chosen horizontal grid resolution, which is 64. Second, the ocean model contains an elliptic equation solver which requires significant numbers of messages (MPI all-reduces) and many iterations during each two-hour timestep. When the problem is partitioned over more than 64 processors, the number of messages generated simply overwhelms the communication ability of the current target machines.

Finally, the PCM is an all message passing application. It is partitioned in a way such that there is a straightforward, one-to-one, mapping of MPI processes to processors. There is no threading on compute nodes that contain multiple processors.

5 Performance

5.1 Productivity

The overall performance as well as productivity of the PCM on its targeted production machines is summarized in Fig. 2, which shows the number of simulated years per wallclock month achieved by a fixed version of PCM over the past several years. Climate modelers prefer to measure model performance in terms of scientific productivity rather than floating point operation counts for at least two reasons: (1) productivity, not floating point counts, is the goal of the scientific investigation, and (2) floating point operation counts are difficult to determine on many machines.

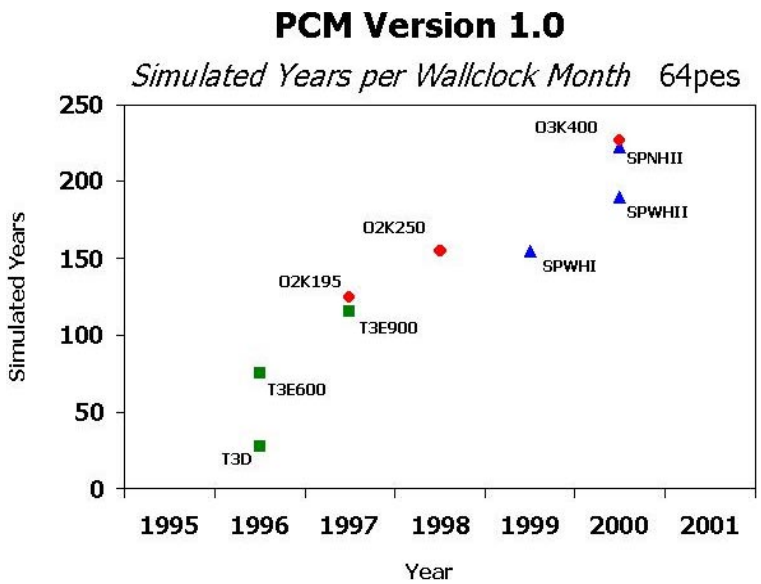


Fig. 2. PCM productivity for a fixed model version on 64pes of various machines from 1995 through 2000. Cray T3D and T3E, SGI Origin 2000 and 3000 (O2K, O3K, with chip clock speed given as last three digits), and IBM SP (WHI is WinterHawk I node, WHII is WinterHawk II node, and NHII is NightHawk II node) are shown.

The original target machine of the PCM was the Cray-T3D, on which the rate of production was one simulated year per wallclock day using 64 processors. With the introduction of the Cray-T3E, especially the T3E-900, the production rate of PCM increased to the same range as other similar climate model codes which were designed for and used on parallel vector machines in production at various U.S. sites in 1997.

In successive years new machines became available. The design of PCM, in particular the use of a standard message passing library (MPI), allowed quick deployment of the model onto these machines. The PCM was given the enviable opportunity of being one of the first users at a number of sites, and the fact that we were able to take significant advantage of these opportunities is witness to the flexible design and portability of the PCM. The scientific returns with PCM for a fixed problem size increased rapidly as new, faster machines were introduced. Our current production rate for a single integration is six simulated years per wallclock day – a six-fold increase in productivity over a five-year period.

5.2 Scalability

The question arises as to how well PCM scales on these machines. Fig. 3 shows the model scaling for a variety of machines. Generally speaking, PCM scales reasonably well up to 32 processors and acceptably well up to 64 processors.

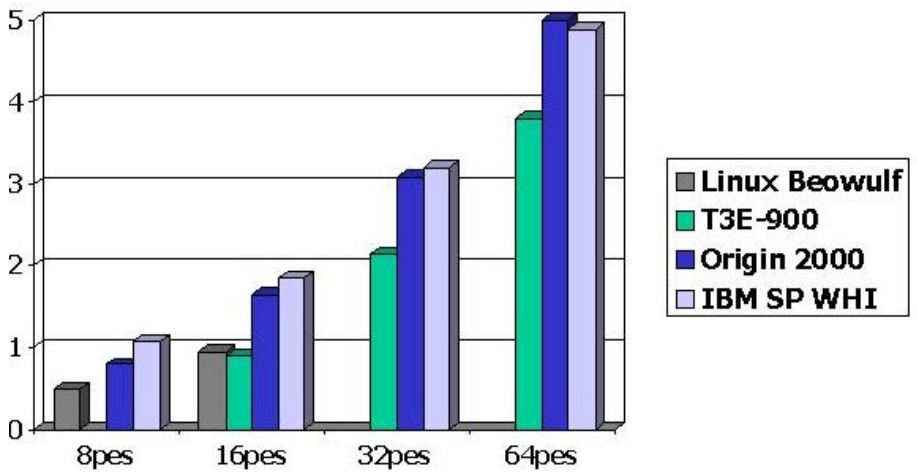


Fig. 3. PCM scalability from 8 to 64 processors on four machines. Simulated years per wallclock day.

As was mentioned in section 4, the PCM is limited inherently to 64 processors due to the internal data decomposition of the atmospheric model. Additionally, the PCM limit of scalability is reached at 64 processors on the current generation of machines due to a combination of (1) the message latency and bandwidth on the hardware (an exception is the Cray-T3E), and (2) the message traffic generated by the model, in particular the elliptic equation solver in the ocean model. With respect to the communication within the components themselves, the individual model developers are exploring new methods to minimize messages, but no attempt has been made to address these issues within the PCM.

6 Port Validation

Use of the PCM across a variety of vendor platforms at a number of diverse computational centers involves validating the fact that the model was ported successfully to a new machine or site. The topic of port validation is important for obvious reasons. From platform to platform the model solutions, obtained using identical code and identical initial conditions, will diverge. *How does one know the answers are correct?*

The portability of PCM is validated by a procedure outlined by Rosinski and Williamson (1997). On a machine with which one has confidence in the solution, two simulations are produced. The second simulation is identical to the first, except that the solution path is forced to be different by introducing a random perturbation, which can be felt by the model at the precision of the machine, into the initial state. The growth of differences between the two solutions is considered to be representative of machine roundoff errors, and is assumed to accumulate only because of the initial, low-order perturbation. By design these errors are typical of the errors expected when the model is moved to a different architecture or when a new or different compiler is used, re-ordering some calculations. The difference of the simulation on the destination machine versus the original simulation on the first machine should exhibit similar characteristics as the differences generated by the two simulations on the first machine. Specifically, the differences during the first few timesteps should be close to the magnitude of machine rounding, and the differences should grow no faster than the differences from the two control simulations on the first machine.

Fig. 4 shows the result of a PCM port validation exercise. Note that the simulation differences, as measured by the RMS difference of surface temperature, between two machines are within one to two orders of machine rounding during the first few timesteps. During the first 144 timesteps (two days), the growth of the difference between the original and ported solutions do not exceed the growth of the initial perturbation introduced into the low-order bits of the original solution. In this case, the port is considered valid.

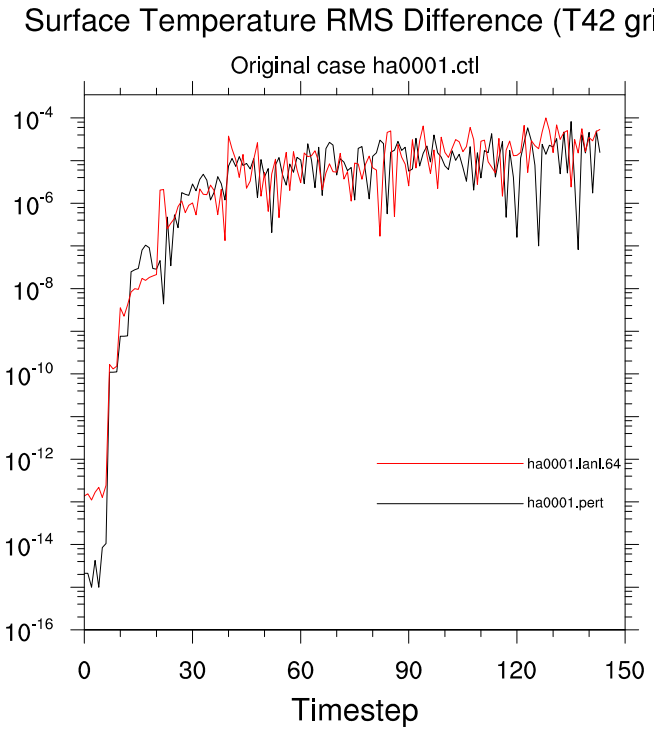


Fig. 4. Port validation of the PCM. The black line is the difference between two simulations on the control machine, which differ only because of an initial, low-order perturbation. The red line is the difference between two identical simulations produced on different architectures.

With climate modeling, it is equally important that the original and ported climate states are statistically identical. This can be accomplished only by making longer integrations and comparing the climates of the two states. This test is also part of the standard PCM validation procedure.

7 Scientific Results

The overall ability of PCM to simulate the earth's climate is demonstrated by the model's ability to simulate the climate during recent past, shown in Fig. 5. Arguably, the model reproduces the observed changes during the past 130 years in the globally averaged surface temperature. This version of the model was forced using observed greenhouse gas values (which act to produce heating at the surface), observed atmospheric sulfate aerosols (which act to produce cooling at the surface), and observed values of the incoming solar energy at the top of the atmosphere (which is observed to have slight variations over time). It is believed that the observed

warming at the surface of the earth is caused by the net effect of changes in the radiative forcing on the earth, of which these three are major contributors.

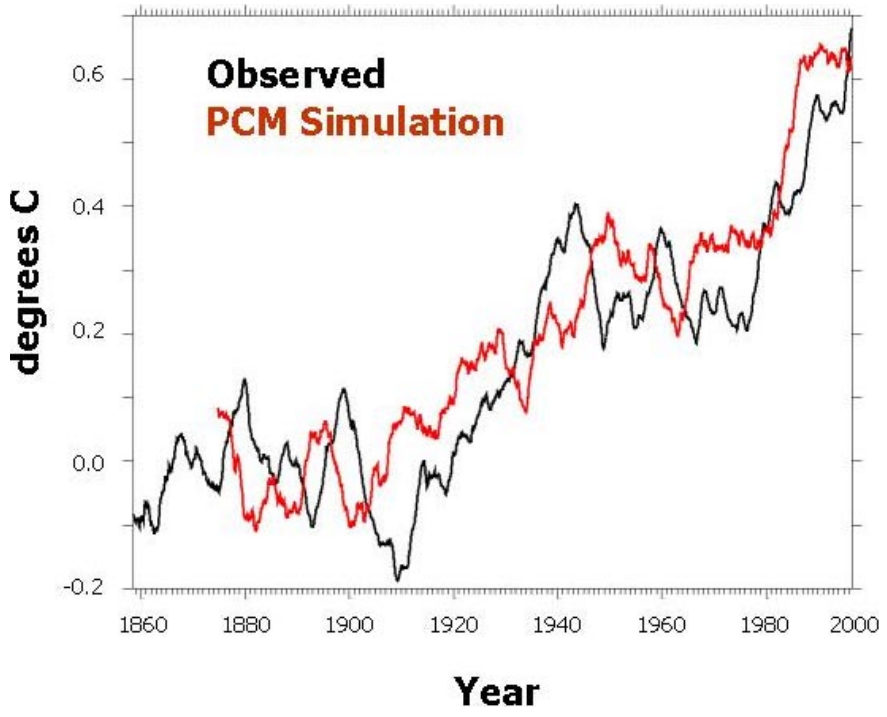


Fig. 5. Global surface temperature anomaly in degrees centigrade for 1860-1999. Observed is the black line. PCM simulation is the red line.

The magnitude of the temperature change, as well as the specific causes responsible for the change, are the subject of continuing scientific debate. In addition, the coarse resolution of the horizontal and vertical grid at the earth's surface is insufficient to allow definitive conclusions to be drawn concerning potential regional climate changes. The answers to questions like these lie, at least partially, in the effectiveness with which scientists can obtain and use increased computational power.

8 Future

At NCAR a climate model similar to the PCM was build with funding from the National Science Foundation – the Climate System Model (CSM) (CSM Model Plan, 2000). The CSM was designed originally and primarily for shared memory, vector parallel, architectures. Over the next year the PCM and the CSM will combine

efforts, and a new model is currently under construction and will contain the following features, which address many of the original PCM shortcomings:

- individual components will integrate in Multiple Instruction Multiple Data (MIMD) mode; the components will run concurrently over the synchronous time period
- all components will contain a general two-dimensional data decomposition
- hybrid parallelism will be implemented within all components (threading on node, and message passing between compute nodes)
- communication between components via the coupler will be generalized and optimized
- specific problems (e.g., the elliptic equation solver in the ocean model) will be addressed with improved, computationally efficient, algorithms

The model should be ready for use in 2002.

References

1. Bonan, G.B.: A Land Surface Model (LSM version1) for Ecological, Hydrological, and Atmospheric Studies: Technical Description and User's Guide. NCAR Technical Note NCAR/TN-417+STR. National Center for Atmospheric Research, Boulder, Colorado (1996)
2. Branstetter, M.L., Famiglietti, J.S., Washington, W.M., and Craig, A.P.: Using a 200-year simulation of a fully coupled climate system model to investigate the role of the continental runoff flux on the global climate system. Climate Variability, the Oceans, and Societal Impacts Symposium, American Meteorological Society, Albuquerque. (2001)
3. Community Climate System Model Plan (CCSM) for 2000-2005. Prepared by CCSM Scientific Steering Committee. National Center for Atmospheric Research, Boulder, Colorado (2000)
4. Cubasch, U., Meehl, G.A., Boer, G.J., Stouffer, R.J., Dix, M., Noda, A., Senior, C.A., Raper, S. and Yap, K.S.: Projections of Future Climate Change. Climate Change 2001: The Scientific Basis. In: Houghton, T, Ding Y., Griggs, D.J., Noguer, M. van der Linden, P., Dai, X., Maskell, K, and Johnson, C.I. (eds.) Contribution of Working Group I to the Third Assessment Report of the Intergovernmental Panel on Climate Change (IPCC). Cambridge University Press (2001)
5. Dukowicz, J.K.and Smith, R.D.: Implicit Free-Surface Method for the Bryan-Cox-Semtner Ocean Model. Journal of Geophysical Research (1994)
6. Jones, P.: First and Second -Order Conservative Remapping. Monthly Weather Review (1999)
7. Kiehl, J.T.: Simulation of the Tropical Pacific Warm Pool with the NCAR Climate System Model. Journal of Climate (1998) 11:1342-1355
8. Rosinski, J.M. and Williamson, D.L.: The Accumulation of Rounding Errors and Port Validation for Global Atmospheric Models. Journal of Scientific Computation, Vol. 18:2 (1997)
9. Washington, W.M., Weatherly, J.W., Meehl, G.A., Semtner, A.J., Bettge, T.W., Craig, A.P., Strand, W.G., Arblaster, J.M., Wayland, V.B., James, R., Zhang, Y.: Parallel Climate Model (PCM) Control and Transient simulations. Climate Dynamics, Vol. 16 (2000) 755-774

Conceptualizing a Collaborative Problem Solving Environment for Regional Climate Modeling and Assessment of Climate Impacts

George Chin Jr., L. Ruby Leung, Karen Schuchardt, and Debbie Gracio

Pacific Northwest National Laboratory, P.O. Box 999,
Richland, Washington 99352 USA
{George.Chin, Ruby.Leung, Karen.Schuchardt,
Debbie.Gracio}@pnl.gov

Abstract. Computational scientists at Pacific Northwest National Laboratory (PNNL) are designing a collaborative problem solving environment (CPSE) to support regional climate modeling and assessment of climate impacts. Where most climate computational science research and development projects focus at the level of the scientific codes, file systems, data archives, and networked computers, our analysis and design efforts are aimed at designing enabling technologies that are directly meaningful and relevant to climate researchers at the level of the practice and the science. We seek to characterize the nature of scientific problem solving and look for innovative ways to improve it. Moreover, we aim to glimpse beyond current systems and technical limitations to derive a design that expresses the regional climate or impact assessment modeler's own perspective on research activities, processes, and resources. The product of our analysis and design work is a conceptual regional climate and impact assessment CPSE prototype that specifies a complete simulation and modeling user environment and a suite of high-level problem solving tools.

1 Introduction

At Pacific Northwest National Laboratory (PNNL), a group of computational and earth scientists have been collaborating on the design of a collaborative problem solving environment (CPSE) for regional climate modeling and assessment of climate impacts. This collaborative effort is at the crosscurrents of two fields of specialty at PNNL. In the area of climate modeling and impact assessment, PNNL atmospheric scientists have developed a regional climate model based on the Penn State/NCAR Mesoscale Model (MM5) that is capable of simulating climate conditions over topographically diverse regions with spatial details down to one square kilometer. The model, PNNL-RCM [7, 9], has been applied to several geographic regions to elucidate the impacts of climate change on water resources and demonstrate the potential use of seasonal climate forecasts for water resources management [6, 8, 10]. In the area of CPSEs, PNNL computational scientists have experience and expertise in the research and development of CPSEs—m ost notable is the Extensible Computational Chemistry

V.N. Alexandrov et al. (Eds.): ICCS 2001, LNCS 2073, pp. 159-174, 2001.
© Springer-Verlag Berlin Heidelberg 2001

Environment (ECCE) [4]. ECCE is a domain encompassing problem solving environment that enables chemists to transparently utilize complex computational chemistry modeling software and access high-performance compute resources from their desktop workstations.

Today, the earth sciences community is intensely focused on the rapid advancement and deployment of climate modeling capabilities as the potential hazards of global warming have been more widely recognized and investigated [15]. The general development approach is to build advanced climate modeling capabilities by leveraging existing modeling and analysis tools as well as high performance computational infrastructures [13]. As such, computational science research among the climate modeling community emphasizes largely on immediate issues such as the development of more sophisticated numerical algorithms, improving or automating the application of models, accessing large amounts of data, and running simulations across distributed networks. The focus is one of near-term, practical concerns as computational scientists incrementally provide advanced scientific simulation capabilities to climate researchers.

Computational scientists at PNNL are following a similar path in the advancement of climate modeling capabilities as we focus on incremental developments such as the parallelization of regional climate modeling codes, enhancement of analysis and visualization techniques and tools, and support for real-time computational steering. At the same time, we are also keeping an eye towards the future by envisioning how climate researchers might collaborate and work in a more ideal computational environment. Where an incremental development approach forces us to couch the climate modeling domain in view of existing computational infrastructures and technologies, the goal of envisionment is to project the advancement of computer technologies and infrastructures to meet the research functions and needs of the climate domain. In essence, envisionment guides development towards an inspiring, future vision. Both incremental and envisionment approaches are important to provide a strategy for climate and computational science to progress towards an optimal medium.

2 Conceptual Prototype Development

The climate CPSE prototype was the product of working with various domain scientists over the course of approximately six months. In the initial phase of our study, we met with five different groups of domain scientists consisting of computational chemists, regional climate modelers, nuclear magnetic resonance experimentalists, automotive engineers, and fluid dynamics modelers [2]. Through a series of participatory analysis sessions, the domain scientists elaborated their basic research and problem solving processes, functions, and needs. Our analysis with domain scientists revealed the following three areas where scientific problem solving could be enhanced through computational support:

- Easy and effective access to computational resources – Scientists need ready access to computational resources such as applications, data, data archives, and computers

in order to run their models. Resources should be represented in a way that is comprehensible and intuitive to the domain scientist.

- Experimental design and execution support – Scientists need to be able to intelligently apply computational resources in the context of a scientific process or experiment. The design and execution of computational experiments is often complex and intricate, and scientists could use better tools to assist them in defining, managing, executing, analyzing, interpreting, and sharing experiments.
- Domain and procedural knowledge management and dissemination – The ability of scientists to solve problems hinge on their knowledge of domain concepts and theories as well as the operational steps required to identify solutions. By making this knowledge explicit and concrete, scientists may be able to better maintain and evolve this knowledge, as well as share it with others.

Following analysis, the conceptual design effort was a modest undertaking involving two software designers, a regional climate modeler, and a hydrology researcher. In both analysis and design, we adhered closely to principles of participatory design [3, 14], which allowed scientists to actively participate in the actual analysis and design work. During several participatory design sessions, the collaborators engaged in the construction of a paper prototype, using large sheets of paper, pencils and pens, color markers, and Post-it™ notes. Through the use of such low-tech tools, the climate researchers were given the capacity to express their needs and ideas in the conceptual design. The paper prototyping technique we employed was based on the PICTIVE (Plastic Interface for Collaborative Technology Initiatives through Video Exploration) design method [12]. The paper prototype was then transformed into a computer prototype.

The scientific content of the conceptual prototype is based on a pilot project funded by the U.S. Department of Energy (DOE) Accelerated Climate Prediction Initiative (ACPI) [13]. Using existing climate modeling and impact assessment tools and capabilities, the ACPI pilot project aims to demonstrate an "end-to-end" approach that begins with the current state of the global climate system and ends with predicted impacts of anthropogenic climate change on water resources at the local and regional levels in the western United States. This approach requires data flow from global climate simulations to regional climate models to various independent assessment models of surface and groundwater, water management, fish habitat, and fire weather. This approach can be extended to integrated assessments of climate change impacts where data flow is also required among the assessment models for horizontal integration of impacts and feedbacks.

In the remainder of this paper, we describe specific features of the regional climate conceptual prototype and relate them to the three aforementioned scientific problem solving themes. We begin by describing CPSE capabilities for designing and executing experiments because the computational experiment is the central concept in the prototype around which other features and capabilities are organized.

3 Experiment Design and Execution

The scientific experiment is the vehicle and framework through which climate researchers attack their research problems and goals. For climate researchers, the practice of designing and executing climate models has certain tendencies and characteristics. Among those we uncovered in our analysis include the following:

- Climate modelers design their experiments by defining sequential steps that utilize the model, observational data, application tools, computers, and other miscellaneous resources.
- The experimentation process is highly repetitive. The climate modeler repeats a cycle of steps that includes modifying the configuration and initial/boundary conditions of the computational experiment, executing the experiment, and evaluating the generated output and its convergence to observed or theorized results.
- As climate modelers conduct computational experiments, they generally perform long sequences of computer operations such as logging into machines, querying for and collecting data from on-line databases and repositories, transferring data files between machines, running applications on distributed computers, capturing experiment output to files, applying translators to convert data formats, and executing analysis and visualization packages on specific data sets.
- In designing and executing their experiments, climate modelers typically maintain the design and execution processes in their heads and in notes placed on loose paper or in laboratory notebooks.

To support the experiment design and execution process, we introduce a variety of experiment management tools that allow climate researchers to better design, organize, manage, execute, and document their experimental research processes.

3.1 Scientific Workflow Execution

To support experiment design and execution in a climate CPSE, we initially focused on deriving a high-level visual representation that climate researchers could construct and utilize to manage their experimental research processes. This objective led to the investigation of scientific workflow representations and visual programming paradigms. As shown in Figure 1, we designed a graphical experiment management representation that is centered on concepts in data flow and flowchart diagrams [16].

The top workflow diagram in Figure 1 presents a high-level view of the steps or tasks of the ACPI project. The arrows of the diagram denote the sequential relationships as well as the data dependencies among the different tasks. As constructed, the diagram presents an execution model for the overall project. As the tasks are performed and completed, their status changes as indicated by the color of the graph nodes. The specific data sets that are passed from one task to another are identified along the arrows of the graph. The workflow diagram provides an abstract view of the design model and the execution state of the experiment.

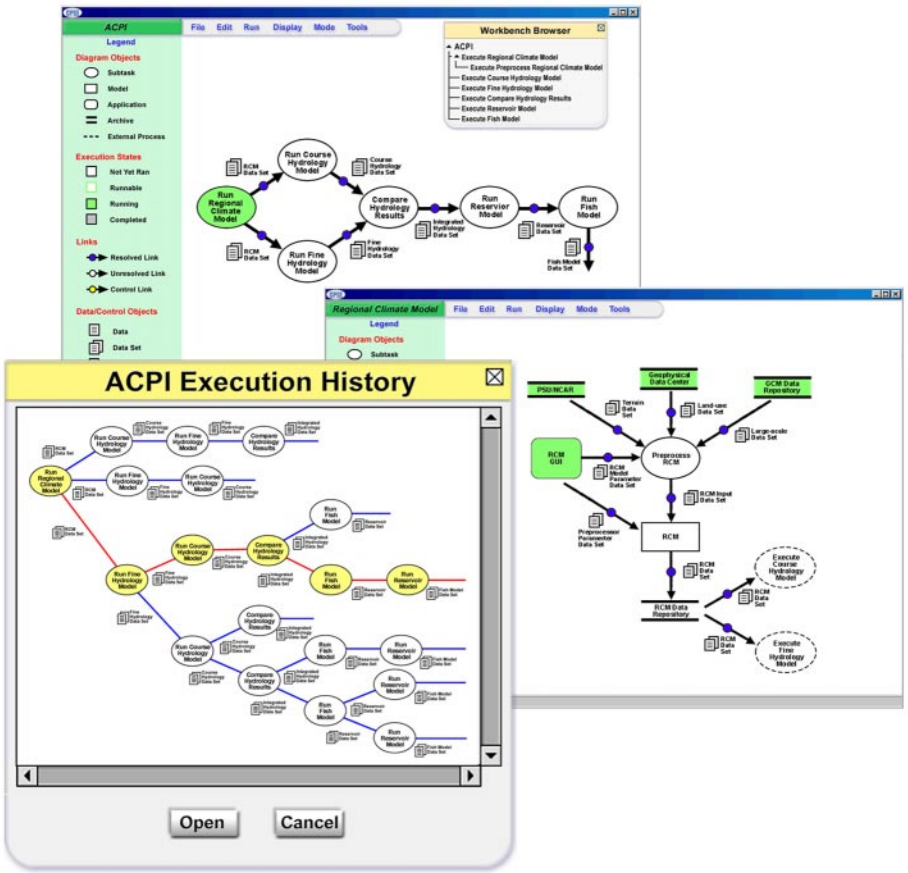


Fig. 1. Scientific workflow diagrams are provided in the climate CPSE prototype to support experimental processes. The two workflow diagrams represent different levels of abstraction. The top workflow diagram depicts the high-level tasks of the ACPI project. The lower workflow diagram expands the Run Regional Climate Model node of the ACPI workflow to expose the specific computational resources that are accessed when the regional climate model is executed. The Execution History tool shows all past executions of a particular experiment. One specific branch in the history tree represents a particular thread of execution.

Graph nodes representing the high-level tasks of a project or experiment may be decomposed into specific experimental and computational resources. For example, in Figure 1, the Run Regional Climate Model task in the top workflow diagram decomposes into the more detailed network of resources shown in the lower workflow diagram. The lower-level resources identify data archives, applications, simulation models, and other subtasks. The higher- and lower-level views of experimental processes represent varying levels of abstraction to the climate modeler. Higher-level views represent more of a scientific and cognitive perspective of the experimental process. They often reflect the experimental steps at the climate researcher's natural level of

thinking. Conversely, lower-level views emphasize the mechanical or procedural steps that are encompassed in scientific tasks.

By encapsulating computational operations within higher-level tasks, the details of the tasks are conveniently packaged for distribution and reuse. The ability to develop multiple layers of abstraction is an important capability for climate researchers because it requires the researcher to specify the intricate operations on resources only once, and then those specifications may be re-used in different runs of the same project, on different projects, or by different researchers. For instance, a novice climate modeler may elect never to expand a specific climate task to view its inner workings, whereas an experienced climate modeler may choose to expand that same task to modify portions of the underlying mechanics to support a different assessment or project.

Figure 2 illustrates the features of the visual execution model. Borrowing concepts from visual debugging, the visual execution model allows climate researchers to set breakpoints and monitors. A breakpoint pauses the execution of the workflow when a particular step is reached. Execution may be resumed from the paused state. A monitor is attached to the output stream of a particular node of the workflow. When data is generated from a monitored node, a specified data viewer or visualization application

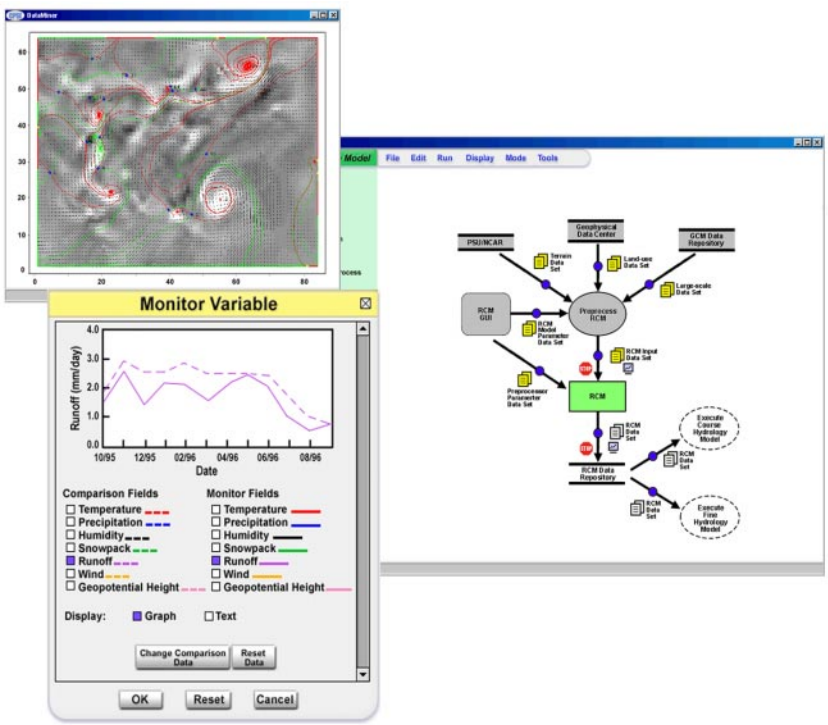


Fig. 2. The workflow execution model provides breakpoints and monitors. Through breakpoints, climate researchers may pause and resume the current workflow execution. Through monitors, visualization tools and data monitors may automatically be spawned once specific output data sets are generated.

is invoked to display some or all of the outputted data.

In our analysis of research processes, we found that climate researchers typically conduct experiments in a dynamic, uneven way. They often pause at critical junctures during the experiment to examine, analyze, or verify critical, intermediate results. During these suspensions, climate researchers may employ one or more data analysis or visualization tools. The development of breakpoints and monitors are intended to support this spontaneous, sporadic mode of experimentation by allowing climate researchers to pause the experiment and conduct on-the-fly analysis and visualization.

For example, in Figure 2, the execution of the PNNL-RCM model and its subsequent generation of the RCM Data Set trigger the invocation of a scientific visualization tool called the DataMiner. In addition to connecting to specific analysis and visualization tools, the climate CPSE prototype also provides a general monitoring tool that allows the climate researcher to select fields of an outputted data set to be tracked and displayed. As shown in the Monitor Variable tool in Figure 2, a simple graphical plot is generated and compared against observed or saved results. Thus, the researcher need not always apply sophisticated analysis or visualization tools to extract basic pieces of information from intermediate experimental results.

3.2 Scientific Workflow Design

An important quality for the visual workflow model is that it be intuitive to and usable by climate researchers and reflects the researchers' views of their own scientific processes. In early analysis sessions, we had climate researchers construct diagrams on paper to convey their research and work practices [2]. The diagrams that they developed were very similar to the workflow diagrams that are applied in the climate CPSE prototype. The fact that the climate researchers were able to effectively construct similar workflow diagrams with relative ease leads us to believe that workflow diagrams may serve as an effective visual modeling approach for climate researchers.

Similar to other component-based visual programming models such as those provided by Advanced Visual Systems AVS [1] and Khoral Research Inc. Khoros [5], the workflow model in the CPSE prototype requires climate researchers to drag and drop modules from a palette of components onto a common workspace and to link the modules using arrows to indicate direction of data flow (see Figure 3). Unlike AVS and Khoros, however, the granularity of the components and processes is at a higher level that deals with computational and experimental resources. The purpose of the workflow model is not to construct a specific application from existing functions or objects but rather to link together a coherent set of resources in a way that allows climate researchers to effectively execute their models and conduct their experiments.

As shown in Figure 3, the climate researcher designs the experiment by placing various resources on the modeling workbench. Along the palette to the left, resources are categorized into scientific models, applications, data archives, files, and encapsulated subtasks. For each resource dragged onto the workbench, the climate researcher is required to specify the resource's attributes, which includes its directory location, run command, run options, host machine, and input and output parameters. For each

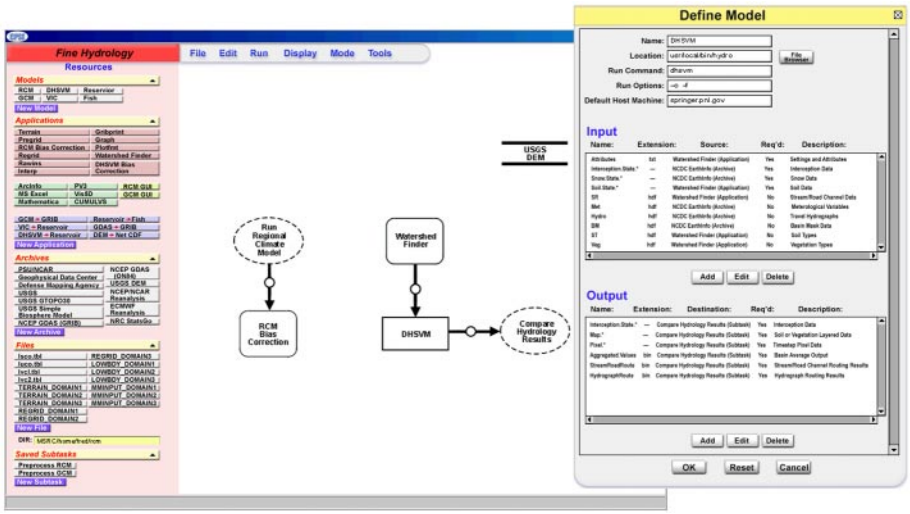


Fig. 3. The climate researcher constructs a workflow model by dragging and dropping resources onto the workbench from the palette. For each resource, the climate researcher must identify its attributes such as the directory location, run command, run options, host machine, and input and output parameters.

link between two resources, the climate researcher must also define the mapping of the data outputs of one resource to the data inputs of another resource.

From our analysis, we found that climate researchers often do not know which applications, data, and resources they will apply until they are in the midst of running their experiments. Thus, the climate CPSE prototype must accommodate the dynamic design and evolution of scientific workflows. The workflow model in the CPSE prototype does not require the climate researcher to fully design and construct a workflow diagram before executing or accessing a computational resource. For instance, the climate researcher may simply drag and drop various resources onto the workbench without connecting them. The researcher may then execute the individual resources separately and manually manage intermediate data among all the resources. This approach is not far removed from the current manner in which climate researchers manually access and apply computational resources. The advantage of performing these operations within the CPSE is that researchers may eventually gain greater comfort and expertise in constructing workflow models over time, but in the short term, they immediately gain additional problem solving capabilities through the other high-level tools provided by the CPSE.

In the climate CPSE prototype, we are also exploring the notion of an adaptable environment that may be customized to the way a climate researcher or group of researchers experiments and collaborates. Monitoring facilities may be built into the user environment such that the environment would keep track of the applications, databases, computers, tools, and other resources that climate researchers apply over time. Based on collected information, the environment may provide automatic links to those resources within the user environment for immediate access. One step further,

the environment may also keep tabs on the specific sequence of actions that particular researchers and groups conduct and automatically construct workflow configurations based on the most common patterns of behavior. This way, climate researchers would not be required to construct workflow diagrams from scratch but could rely on the user environment to provide preliminary workflow models.

3.3 Historical Experiment Execution Record

For any climate experiment, climate researchers make numerous runs against the climate model as they investigate different climate properties, regions, and time periods. In various cases, researchers may wish to revisit previous runs in an effort to reuse the initial conditions or configurations of those runs, track down a particular error or anomaly, reproduce a previous result, or identify patterns or trends in the results. To support these functions, the climate CPSE prototype has defined automated facilities for capturing, managing, and re-instantiating historical records of previous experimental runs.

As shown with the Execution History tool in Figure 1, the climate researcher may highlight and re-instantiate particular paths of executions of the climate model. The graphical tree-based depiction presents a hierarchical view of past executions with nodes identifying points of iteration in the experiment. When a particular path branches out into multiple paths, the multiple paths indicate that several experimental runs were carried out from a common state in the experiment. As shown in Figure 1, the climate researcher may select an extended thread through the history tree to reproduce that particular experimental run. Additionally, the climate researcher may retrieve input and output data sets generated at any point in the execution history.

4 High-Level Access to Computational Resources

Climate researchers do not naturally think of computational resources as applications, computers, and files but rather as models, calculations, and spatial and temporal data. One of our objectives in designing a climate CPSE is to promote the appropriate level of abstraction such that climate researchers may utilize these resources in a form consistent with their specific domain concepts and views. For the most part, we have found that current climate researchers are intimately familiar with the details of the computational environment in which they run their models. Such a prerequisite, however, is a high barrier that may prohibit many researchers, who may have the knowledge and skills to develop climate concepts and models, from contributing due to their inability to navigate the complicated computational environment. In a more ideal world, the computational environment would not be a barrier to conducting valuable climate research, and any climate researcher with sound domain expertise should be able to apply their science and contribute to the expanding body of knowledge.

In the climate CPSE prototype, we have derived a variety of tools designed to provide climate researchers with high-level access and a high-level view of computational resources. Together, these resource tools allow climate researchers to perform what

we call "computational problem solving." In contrast to scientific problem solving, the emphasis of computational problem solving is not based on the science but rather on the effective and in situ operation of computational resources that is required in applying the science. We now describe a few of the tools in the climate CPSE prototype that fall under this category.

4.1 Calculation Table

Climate researchers think about their experiments and models in terms of the calculations they need to run. Prior to executing a climate model, the researcher organizes his investigation as a set of calculations that may be partitioned into different combinations of geographical regions, time periods, earth properties, and parameters for physical processes. Defining a set of calculations allows the climate researcher to break down the experiment into smaller, more manageable pieces and be able to allocate the computational work across distributed and parallel machines. Furthermore, the set of calculations may also serve to identify and manage the division of labor among collaborating researchers. Thus, a group of collaborating climate researchers might partition the set of calculations among group members as individual tasks.

In the CPSE prototype, the Calculation Table allows climate researchers to define, view, and execute a set of calculations while hiding as many of the details as possible of the underlying computer infrastructure and network. As shown in Figure 4, the Calculation Table maintains a dynamic list of climate calculations that are identified by key properties such as geographical location, grid size and resolution, and time period of simulation. Operational properties of the calculations are also maintained such as the machine on which the calculation is to run, the time of scheduled execution, and the project member who is assigned to execute and manage the calculation. Information is inserted into the Calculation Table through a calculation entry form, or the Calculation Table may be fed by a separate graphical user interface (GUI) application that is tied to the climate model. In the case of the latter, the Calculation Table stands as an intermediary between the GUI and the climate model—diverting model input parameters into the Calculation Table for follow-up execution.

4.2 Spatial and Temporal Data Browsers

Current climate models produce large, multi-file data sets that may be in the order of terabytes to petabytes in size. In applying the data, climate modelers must deal with intricate details such as accessing data archives and systems, retrieving large volumes of data, collecting sets of files into an organized view, transferring or streaming data to different locations, and converting both standard and application-specific data formats. Thus, climate researchers currently work with climate data from the view of data formats, data organization across files, file systems, and network topographies. With the climate CPSE prototype, we sought to shift the climate researcher's view of data from a structural or mechanical one to a more conceptual one.

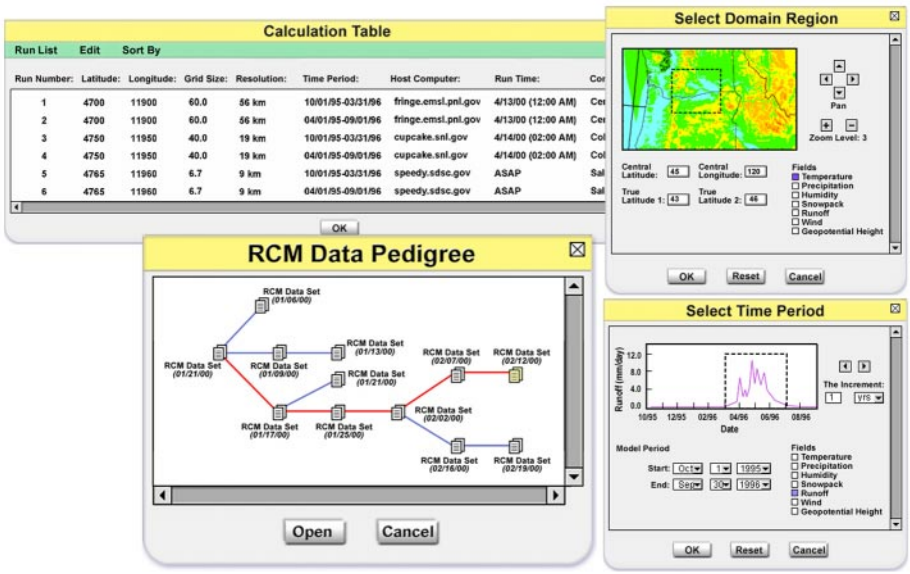


Fig. 4. Climate CPSE tools provide high-level access to computational resources. The Calculation Table manages a list of calculations for a particular experiment. The Spatial Data Browser and Temporal Data Browser allow climate researchers to select data based on spatial and temporal properties. The Data Pedigree tool allows climate researchers to view and access the historical lineage of a data set.

Consistent with the way they define their calculation sets, climate researchers view and apply climate data largely along dimensions of time, space, and earth properties. In addition, the selection of data is usually preceded by some high-level analysis of the data space. To accommodate these usages, the conceptual prototype supports two types of data views as shown in Figure 4. From a geographical or spatial perspective, the Spatial Data Browser allows climate researchers to view, select, and manipulate data and data files in 2D space by selecting regions of a map or image. As directed by climate researchers, a spatial map is overlaid with information, attributes, or contours that illustrate different earth properties. The climate researcher examines features and patterns in the map to identify regions from which the actual data should be extracted. This extraction of data is particularly important for impact assessments as they are typically performed over smaller regions that are defined by climate regimes, watersheds, basins, eco-regions, or farms. From a temporal perspective, the Temporal Data Browser allows climate researchers to view, select, and manipulate data and data files from different parts of a timeline. The climate researcher graphs the values of different earth properties over time. Based on an analysis of the graph, the climate researcher may select a time period over which climate data should be retrieved.

4.3 Data Pedigree Management

As we previously described, the Execution History tool in the climate CPSE prototype allows the climate researcher to navigate through previous instances of models, applications, and other computational resources. Similarly, the climate researcher may also wish to navigate through and revisit previous versions of a data set. The climate researcher evolves and iterates a data set over the course of experimentation. The ability to manage large, complex data sets becomes more difficult over time as more experiments and runs take place. Thus, an automated tool for managing historical versions of data sets would provide a valuable service.

As shown in Figure 4, we introduce the concept of a Data Pedigree tool that provides data set versioning capabilities. The Data Pedigree tool provides a hierarchical tree depicting the "pedigree" or the historical lineage of a climate data set. Each branch in the tree represents a particular path of evolution for the data set. From the tree view, the climate researcher may select a particular node to access a specific version of the data set. Alternatively, the climate researcher may select an entire branch and apply various comparison functions to evaluate how a data set may have changed over time. Overall, the Data Pedigree tool will be useful to researchers for diagnosing the source of an error or anomaly in a data set by providing access to the data set's full history.

5 Knowledge Management and Dissemination

Scientific problem solving is inherently a collaborative effort among researchers as they share information, models, tools, resources, and results. More than just sharing specific research artifacts, however, scientific problem solving also involves the sharing of one's expertise and experiences. As climate researchers run models, they apply a vast amount of procedural and domain knowledge. Climate researchers may have much experience in running particular climate models—understanding the nuances and problems of those specific models. Researchers may also have a lot of knowledge of climate concepts and theories that serves as the basis of their research and experimentation. The ability to capture these kinds of knowledge and share them with others is yet another potential enabling feature of a climate CPSE.

To a certain extent, climate researchers already have mechanisms in place for capturing procedure and domain knowledge. Depending on the climate researcher's particular research style and specific experimentation needs, the researcher may jot rough notes down on a notepad or place long, detailed descriptions in a laboratory notebook. In the climate CPSE prototype, we have provided additional, more advanced knowledge capture and dissemination mechanisms.

5.1 Electronic Laboratory Notebook

As shown on the left side of Figure 5, an Electronic Laboratory Notebook [11] may be used as a knowledge capture tool much like a physical laboratory notebook is applied.

The climate researcher may store theoretical and procedural information in the Electronic Laboratory Notebook during the course of a project. The amount and level of detail of the information stored is under the control of the climate researcher.

An electronic version of the laboratory notebook may have additional features that allow the climate researcher to better manage, locate, and share scientific information. For instance, an electronic version may extend the capabilities of a laboratory notebook by allowing climate researchers to capture information in many different forms (e.g., text, drawings, tables, images, audio files, video files), share content by allowing researchers to access the same notebook from multiple machines, utilize standard security and authentication mechanisms to ensure valid access, and employ indexing and searching mechanisms that allow the researchers to easily and quickly locate specific information. These are among the capabilities that make Electronic Laboratory Notebook attractive as an integrated knowledge management tool.

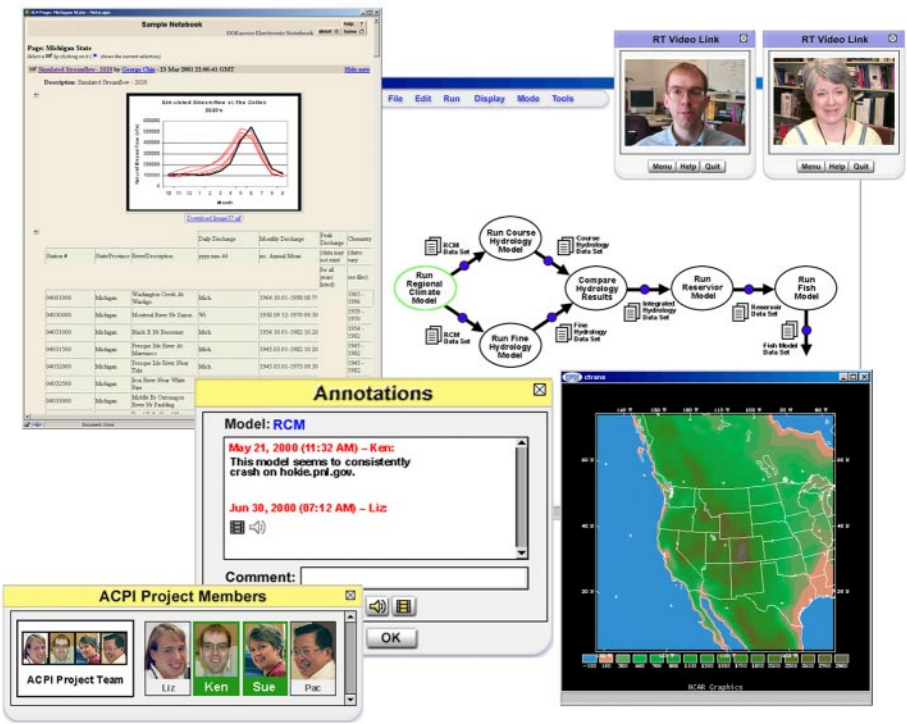


Fig. 5. A dynamic collaboration environment allows real-time collaboration to take place within the context of the climate researcher's experimental process. In the above scenario, the climate researcher works within the climate CPSE, using a workflow and visualization tool. When the climate researcher requests collaboration with project members, the collaboration environment establishes video connections with on-line project members and provides immediate sharing of the workflow and visualization windows. The collaborators may also bring up shared notes in the Electronic Laboratory Notebook and through the Annotation tool.

5.2 Free-Form Annotation

As climate researchers conduct their experiments, they commonly jot down miscellaneous notes pertaining to their current research state such as the line of investigation being followed, the input parameters to the current experiment, key findings and results, and encountered error conditions. The notes or comments that are made may have a temporal, spatial, or referential context. For instance, they may be associated with a particular data set, a phase or step of the experiment, a specific result of the experiment, or a particular resource such as a computer or data archive. In the climate CPSE prototype, many of these associated entities have visual representations. A free-form annotation mechanism would allow climate researchers to enter annotations and link them directly to the their visual referents.

As shown in Figure 5, the Annotation tool allows climate researchers to annotate any design or resource representation. Annotations for any particular object are collected over time. Besides text, we also envision that climate researchers may also create audio and video annotation entries. Once an annotation entry has been created, collaborators with authorized access may view or replay the annotation. The annotation mechanism provides a means for capturing and sharing contextual knowledge. A climate researcher may review the annotations of other collaborators to comprehend the concepts, usage, or problems associated with any particular entity. Through the annotation mechanism, contextual knowledge is embedded into the CPSE and expands over time as more annotations are entered. In due course, the CPSE matures—gaining more knowledge and experience through continual use.

5.3 Computer-Mediated Collaboration

Scientific collaboration does not occur in isolation but is driven by the functions of the scientific research. From our analysis and design studies, we found that climate researchers collaborate at key points during the research process to define the scientific problem and hypotheses to attack, partition research tasks, compare and analyze results, and conduct other expected collaborative activities. Climate researchers also collaborate at random points during the research process to troubleshoot problems, seek additional expertise, compare and validate intermediate results, and carry out other collaborations that are necessitated by unforeseen events occurring during the research. In both situations, collaboration is a natural and dynamic function of the research activity—not an independent, preplanned act.

In the climate CPSE prototype, collaboration is instantiated in the context of the project and the experiment. For example, in the scenario depicted in Figure 5, a climate researcher executes a particular climate model and brings up a visualization of intermediate results. Seeing an unexpected result, the climate researcher initiates a real-time discussion with other on-line project members—video conferencing starts and the executing workflow and the visualization tool become collaborative. During the discussion, collaborating researchers share notes through the Annotation tool and parameters, procedures, and results through the Electronic Laboratory Notebook.

When the conversation ends, the collaboration terminates, and the initial climate researcher returns to previous work.

As illustrated, collaboration capabilities are initiated from within the CPSE rather than from a separate, non-related environment. The collaboration tools and sessions are instantiated rapidly, dynamically, and within the context of the research activities.

6 Summary

The climate CPSE we have described in this paper is currently in the form of an early prototype with very limited functionality. It will be incrementally developed in conjunction with an underlying scientific computing infrastructure using a combined top-down, bottom-up development strategy. The strategy involves simultaneously evolving user and system capabilities until these capabilities converge to a unified, fully functional system. Emerging scientific computing infrastructure capabilities include services for scientific data transport and management, intelligent job launching and control, application integration, security and authentication, and event management. The overall two-pronged development approach is mutually beneficial, because the high-level prototype provides user and system requirements to the infrastructure development effort while the infrastructure work provides features and imposes technology constraints on the development of high-level features and tools.

Climate researchers live in a world of increasing complexity. With more advanced models, faster, more powerful computers, and higher-capacity data archives—all distributed across high-speed networks—researchers have a growing capability to run more sophisticated simulations and solve more challenging scientific problems. Unfortunately, many climate researchers are unable to harvest the vast computational power afforded to them. Rather, the capability is relegated to those select few who are able to comprehend and navigate the complicated computational environment.

In conceptualizing and designing a climate CPSE, we aim to abstract the science of climate modeling and impact assessment as much as possible from the underlying computational infrastructure. We wish to pierce the technological barrier by allowing those who are familiar with the computational environment the capability to execute their experiments more productively and efficiently while providing those who are not the very opportunity to exercise their scientific abilities using advanced capabilities and the latest inventions. For us, the operative term in the creation of CPSEs is "problem solving." Our ultimate goal is to provide climate researchers with greater ability to solve their scientific problems. Thus, beyond allowing climate researchers to execute their models easier and faster, CPSEs should meaningfully and fundamentally transform and enhance the science and the practice of climate modeling and impact assessment as well as energize and empower an entire community of climate researchers to achieve greater scientific heights.

References

1. Advanced Visual Systems: AVS/5. Advanced Visual Systems, Waltham, MA 02154, USA (2000)
2. Chin, G., Schuchardt, K., Myers, J., Gracio, D.: Participatory Workflow Analysis: Unveiling Scientific Research Processes with Physical Scientists. In Proceedings of Participatory Design Conference (PDC 2000), (Nov. 28-Dec. 1, New York, NY), CPSR, Palo Alto, CA (2000) 30-39
3. Greenbaum, J., Kyng M.: Introduction: Situated Design. In Greenbaum, J., Kyng M. (eds.): Design at Work: Cooperative Design of Computer Systems. Lawrence Erlbaum Associates, Hillsdale, NJ (1993) 1-24
4. Jones, D.R., Gracio, D.K., Taylor, H., Keller, T.L., Schuchardt, K.L.: Extensible Computational Chemistry Environment (ECCE). In Fayad, M.E., Johnson, R. (eds.), Domain-Specific Application Frameworks: Frameworks Experience by Industry. Wiley & Sons, New York (2000)
5. Khoral Research Inc.: Khoros Pro 2000 Version 2. Khoral Research Inc., Albuquerque, NM 87110-4142, USA (2000)
6. Leung, L.R., Ghan, S.J.: Pacific Northwest Climate Sensitivity Simulated by a Regional Climate Model Driven by a GCM. Part I: Control Simulation. J. Clim. 12 (1999) 2010-2030
7. Leung, L.R., Ghan, S.J.: Parameterizing Subgrid Orographic Precipitation and Surface Cover in Climate Models. Monthly Weather Review, 126(12) (1998) 3271–3291
8. Leung, L.R., Hamlet, A.F., Lettenmaier, D.P., Kumar, A.: Simulations of the ENSO Hydroclimate Signals over the Pacific Northwest Columbia River Basin. Bull. Amer. Meteorol. Soc. 80(11) (1999) 2313-2329
9. Leung, L.R., Wigmosta, M.S., Ghan, S.J., Epstein, D.J., Vail, L.W.: Application of a Subgrid Orographic Precipitation/Surface Hydrology Scheme to a Mountain Watershed. J. Geophys. Res. 101 (1996) 12803-12817
10. Leung, L.R., Wigmosta, M.S.: Potential Climate Change Impacts on Mountain Watersheds in the Pacific Northwest. J. Amer. Water Resour. Assoc. 35(6) (1999) 1463-1471
11. Mendoza, E.S., Valdez, W.T., Harris, W.M., Auman, P., Gage, E., Myers, J.D.: EMSL's Electronic Laboratory Notebook. In Proceedings of the WebNet '98 World Conference, (Nov. 7-12, Orlando, FL), (1998)
12. Muller, M.J., Wildman, D.M., White, E.A.: 'Equal Opportunity' PD Using PICTIVE. Communications of the ACM, 36(4) (1993) 54-66
13. Patrinos, A.: The Accelerated Climate Prediction Initiative: Bringing the Promise of Simulation to the Challenge of Climate Change, [Online report]. Available URL: http://www.epm.ornl.gov/ACPI/Documents/ACPIfinal.html, (1998)
14. Schuler, D., Namioka, A. (eds.): Participatory Design: Principles and Practices. Erlbaum Associates, Hillsdale, NJ (1993)
15. US Global Change Research Program: The Potential Consequences of Climate Variability and Change: Foundation Report. Cambridge University Press (2000)
16. Yourdon, E.: Modern Structured Analysis. Prentice Hall, Englewood Cliffs, NJ (1988)

Computational Design and Performance of the Fast Ocean Atmosphere Model, Version One

Robert Jacob[1], Chad Schafer[2], Ian Foster[1], Michael Tobis[3], and John Anderson[3]

[1] Argonne National Laboratory, Mathematics and Computer Science Division
9700 S. Cass Ave., Argonne, IL 60439, USA,
jacob@mcs.anl.gov
[2] University of California at Berkeley, Department of Statistics
367 Evans Hall, Berkeley, CA 94720, USA
[3] University of Wisconsin–Madison, Space Science and Engineering Center
1225 W. Dayton St., Madison, WI 53706, USA

Abstract. The Fast Ocean Atmosphere Model (FOAM) is a climate system model intended for application to climate science questions that require long simulations. FOAM is a distributed-memory parallel climate model consisting of parallel general circulation models of the atmosphere and ocean with complete physics paramaterizations as well as sea-ice, land surface, and river transport models. FOAM's coupling strategy was chosen for high throughput (simulated years per day). A new coupler was written for FOAM and some modifications were required of the component models. Performance data for FOAM on the IBM SP3 and SGI Origin2000 demonstrates that it can simulate over thirty years per day on modest numbers of processors.

1 Introduction

The Earth's climate is the long-term average of the behavior of the ocean, land surface, and sea ice. These components exchange heat, momentum, and fresh water with each other through their common interfaces. The difference in heat storage times, fluid dynamic scales, and transport/storage of water results in complex interactions between the components.

To study the Earth's climate, researchers have developed coupled models, comprising an atmospheric general circulation model (GCM), a model of the ocean general circulation, a model of the dynamic and thermodynamic properties of sea ice, and a model of the temperature and composition of the land surface. Typically, each component model is a separately developed program with its own code style, language, data structures, numerical methods, and associated numerical grids. Constructing a single model from these submodels requires several design decisions about grid structure, model decomposition, and flux calculations. Additionally, a "coupler" must be created to enable the subcomponents to exchange heat and other quantities.

V.N. Alexandrov et al. (Eds.): ICCS 2001, LNCS 2073, pp. 175–184, 2001.
© Springer-Verlag Berlin Heidelberg 2001

While many coupled climate models are used in studies that require predicting fine-scale details of temperature and precipitation changes, other climate science studies are more concerned with high throughput than with high resolution. For example, paleoclimate scientists wish to know how the climate responded to conditions thousands of years ago, when the earth received more solar radiation in the summer or when glaciers covered much of the northern hemisphere; and climate scientists wish to know how much low-frequency internal variability the climate system has and how it might interact with anthropogenic change. Integrating a climate model to even a quasi-equilibrium for the myriad of interesting paleoclimate scenarios requires hundreds of simulated years, while studying low-frequency variability requires simulations of thousands of years. Thus there is a compelling need for a climate model with high-throughput–a high number of simulated years per day.

The Fast Ocean Atmosphere Model is a coupled climate model designed to meet this need. It has already been used to examine low-frequency variability and paleoclimate problems [7,11,13]. This paper describes how the goal of high-throughput guided the design of FOAM and its coupler. Section 2 describes the component models of FOAM. The unique elements of FOAM's software design are described in Section 3. In Section 4, performance data for FOAM is presented in the form of throughput per processor for various platforms. Section 5 briefly outlines future work on the next version of FOAM.

2 Component Models

The first design decision made for FOAM was to use distributed-memory parallel components. The low cost/performance ratios of distributed-memory multicomputers make them the platform of choice for the kind of climate science questions FOAM is intended to address. Fortunately, FOAM was able to use several existing parallel component models, as briefly described below. (A detailed description of the physical equations solved by each component is beyond the scope of this paper; however, that information can be found in the cited references.)

The atmospheric component of FOAM is derived from PCCM2, developed by Argonne National Laboratory, Oak Ridge National Laboratory, and the National Center for Atmospheric Research. PCCM2 is a functionally equivalent version of NCAR's CCM2 in which the physics and dynamics calculations have been given a two-dimensional parallel decomposition (see Figure 1a for an example). The basic equations and numerical methods used in CCM2 (Eulerian-spectral elements for the dry dynamics and semi-Lagrangian transport for moisture) are described by Hack et al. [6], and the alterations necessary to implement a parallel version and its performance on various parallel platforms are described by Drake et al. [5]. When CCM3 [8] was released, the FOAM development team added the new physics parameterizations to the atmosphere model to create the current atmosphere component of FOAM, informally called PCCM3-UW.

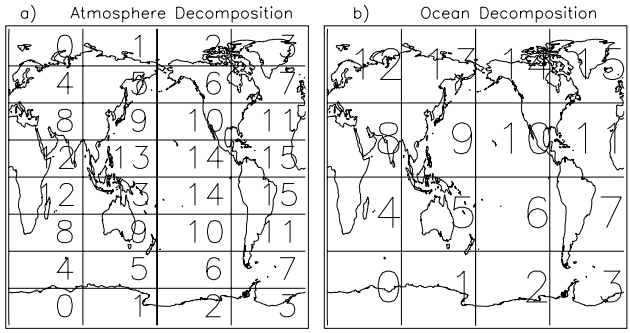


Fig. 1. A depiction of how physical space is partitioned among 4x4 processors according to the decomposition strategy of (a) the atmosphere and (b) the ocean.

The ocean component of FOAM, called OM3, is also a parallel model with a two-dimensional decomposition (Fig. 1b). OM3 is a z-coordinate finite-difference model. While documentation specific to OM3 is still under development, the basic equations solved are the same as those for the widely used z-coordinate Modular Ocean Model created by GFDL [4] [3]. However, FOAM uses a free surface as described by Killworth et al. [9] and contains numerical methods specifically chosen for their efficiency on distributed-memory parallel platforms [14].

The land surface and sea-ice models in version one of FOAM are based on those of PCCM2 [6] but with some important modifications. The prescribed evaporation and snow cover have been replaced with a prognostic box hydrology model from CCM1 [15]. Also, the effect of sea-ice creation/destruction on ocean salinity has been included.

Two new software components had to be created to complete FOAM. The coupler will be described below. The other new component is a parallel river transport model. In order to prevent a constant increase in salinity of the ocean in long climate simulations, it was necessary to close the hydrologic cycle by returning precipitation that falls on land to the ocean. FOAM's river model is a parallel implementation of concepts described by Miller et al. [12].

FOAM and its individual components use the Message Passing Interface (MPI) library for communication. The land and sea-ice models do not require any communication, however, since they are implemented as one-dimensional models at each land and ocean point on the surface, respectively.

3 Design of the Coupled System

In parallel climate models, each component, especially the atmosphere, can place large demands on the communication network and CPU resources of modern parallel supercomputers. A coupled system creates additional demands on the

bandwidth (mostly from the exchange of data between models) and on CPU resources (mostly from the interpolation of data between numerical grids). Design choices in FOAM were made to minimize the impact of coupling. Those choices and the reasons for them are summarized in this section.

3.1 Number of Numerical Grids

Generally, each model may have its own finite difference grid. FOAM's design starts with the requirement that all surface models (land, sea ice, and ocean) use the grid of the ocean model. The intention is to eliminate a class of operations in the coupler that would be required to keep track of land/ocean fraction and merge the calculated fluxes accordingly. This decision does not have a large impact on the performance of the model, but it does greatly simplify the conceptual picture of the coupled system. It also has implications for other design choices described below.

3.2 Decomposition of Component Models

Just as each model may have its own numerical grid, each parallel model may decompose that grid in a different way over its set of MPI processes. Additionally, each model may have physically separate pools of processors. In order to calculate a new flux or provide to a model an internally calculated flux, such as precipitation, data that occupies the same physical space must be brought into the same physical piece of memory. The bandwidth cost of this communication will depend on how dissimilar the decompositions are and how much overlap there is of model state in memory. To minimize this cost, FOAM restricts the possible decomposition of some of the component models based on a consideration of the time scales between components, as shown in Figure 2. In general, an

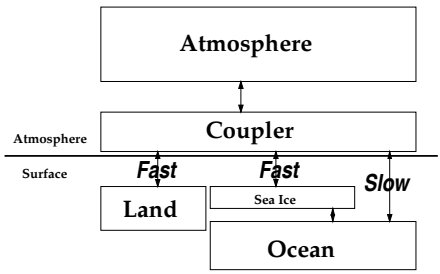


Fig. 2. Schematic of the components of FOAM indicating which components interact on fast and slow time steps and reflecting the physical relationship between components. The parallel river model is omitted for clarity.

atmosphere model has a shorter basic time step than does the ocean because of

differences in the dominant fluid dynamical scales in the flows. In FOAM, the atmosphere has a basic time step of 30 minutes, while the ocean uses a 6-hour time step. Because of the small heat capacity of land and sea ice, their temperature structure varies with the diurnal cycle of solar heating (which is resolved by the atmosphere model), and thus they need to communicate with the atmosphere every one or two time steps. The ocean, however, with a much larger heat capacity, does not have a significant diurnal cycle and needs to communicate with the atmosphere only once a day.

In order to reduce the costs of communicating fluxes, the "fast" components (Fig. 2) are all given the same decomposition so that there is nearly a one-to-one correspondence in area coverage between atmosphere, land, and sea ice. The atmosphere, land, and sea-ice models are executed sequentially on the same pool of processors so nearly all data necessary for physical coupling resides in local memory. The area coverage will not match exactly, however, because the land and sea ice are on the ocean's numerical grid, which coincides nowhere with the atmosphere's. Some communication is required to update points on the edges of the local rectangular regions covered by each MPI process, but this communication overhead is less than what would be required if the fast components had to transfer or transpose their entire state across multiple processors every one or two time steps.

The ocean model has its own decomposition and is assigned a distinct set of processors. It integrates concurrently with the atmosphere, land, and sea ice. Trial and error determines how many processors to allocate to each side to avoid blocking on data. Communication between the ocean and atmosphere–land–sea-ice coupler is done by designating one MPI process on each side to receive all the data from the other component and redistribute it according to that component's decomposition strategy. Since the ocean is the slow component and needs to communicate less frequently, the impact of this single-node bottleneck is minimized.

3.3 Flux Interpolation and Calculation

One of the basic tasks of a coupled model is to interpolate quantities from one grid to another. Calculating and interpolating fluxes in FOAM are simplified by placing the land, sea-ice, and ocean models all onto a common grid as mentioned above. Thus, the number of grids in the model is reduced to at most two instead of possibly one per model.

FOAM considers the surface of the globe as being divided into rectangles or tiles. The center of each tile is at a computational point of the model, and the four edges are halfway between each of the neighboring computational points. A third set of tiles is made by laying the ocean grid on top of the atmosphere grid, as in Fig. 3a. New fluxes are calculated on this overlap grid set of tiles. The flux through a model tile is constructed by area-weighted averaging the appropriate pieces of the overlap tiles as shown in Fig. 3b. Global conservation is assured as long as all the overlap pieces are used once and only once in the averaging calculation for each grid.

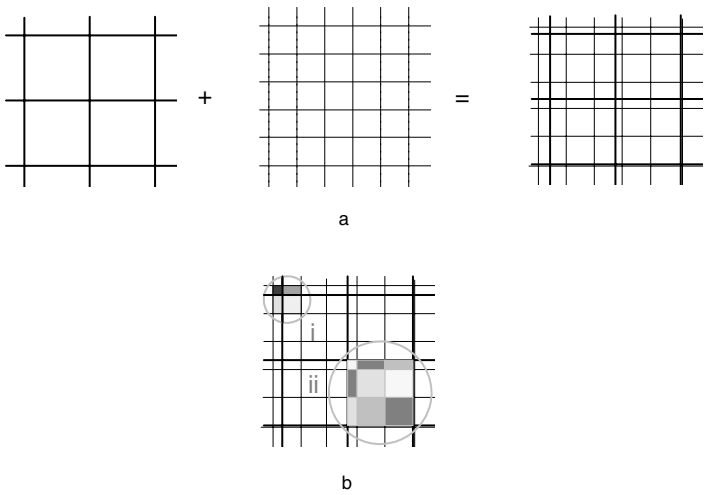


Fig. 3. (a) Forming the overlap grid. (b) A surface (region i) and atmosphere tile (region ii) composed of overlap pieces.

The interpolation method can be demonstrated further by considering the calculation of sensible heat. This requires the temperature of the lowest atmosphere level TA and a surface temperature TS. FOAM's calculation of the sensible heat flux through an overlap tile can be approximated as

$$SH = K(TS_{i,j,k} - TA_{l,m}), \tag{1}$$

where K is a constant. The triplet i, j, k indexes the pieces a given surface tile i, j is divided into (1, 2, or 4 pieces), and l, m indexes the atmosphere tile lying above the piece. The mapping of the overlap tile indices onto each of the two grids and the area of the overlap tiles are calculated offline and stored in lookup tables for use during runtime. Once the flux at each overlap tile is known, an area-weighted aggregate is formed for each ocean and atmosphere tile in the subdomain of the coupler, as illustrated in Fig. 3b.

3.4 FOAM's Coupler

After all the simplifications made with component models, FOAM's coupler is relatively straightforward. The coupler is implemented as a subroutine of the atmosphere. The coupler performs all the communication necessary to resolve the incomplete overlapping of the local surface and atmosphere grids and calls the land, sea-ice, and river models (which are themselves subroutines of the

coupler). The merging of fluxes coming from separate ocean, land, and sea-ice models into a single atmospheric cell is greatly simplified by the use of single surface grid. Finally, the coupler accumulates ocean fluxes and communicates with the ocean as necessary. "Flux corrections", which are sometimes needed to achieve a stable coupled climate, are not used in FOAM.

3.5 Constructing the Full Model

The noncoincidence of the two grids presented a difficulty when combining the coupler and fast surface models with the atmosphere. In the original atmosphere model, PCCM2, all physics routines were called once for each latitude, with the surface physics occurring about two thirds of the way through the sequence. The uneven overlap of the two grids means the coupler occasionally needs two atmosphere latitudes to calculate the flux through one ocean latitude. Consequently, the calling tree for the PCCM2 physics was split so that all the physics before the surface package could be completed for the whole globe before calling the coupler and surface models. The atmosphere resumes execution when the coupler returns. This split is still present in FOAM's current atmosphere model PCCM3-UW. (NCAR's CCM3, which unlike PCCM2 was designed to be part of a coupled system, contains a similar split [1].)

FOAM is a single executable image. A small main program divides processors between the atmosphere-coupler-surface component and the ocean according to compile-time settings and then calls each. Some minor modifications to the atmosphere and ocean were necessary to turn them into subroutines. A summary of FOAM's structure is shown in Fig. 4.

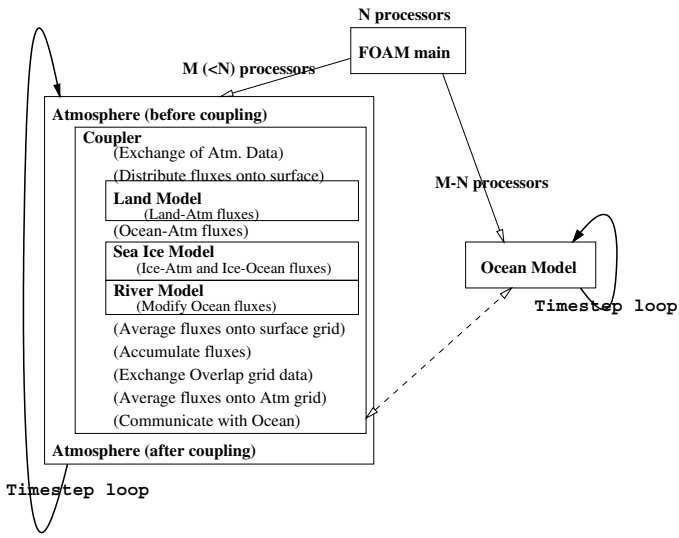


Fig. 4. A schematic of FOAM's software design.

The atmosphere and ocean are each told by the main routine how many days to integrate. The atmosphere directs the execution of itself, the coupler, and the surface components that share the same processors. The ocean model waits to receive fluxes from the coupler and send sea surface temperatures according to a frequency set at compile time (currently every ocean time step).

4 Model Performance

The performance of FOAM and some of its components is presented in Figure 5. While the atmosphere model supports many choices for truncation level of the spectral transform numerical method, the standard configuration for FOAM uses a rhomboidal truncation with fifteen wave numbers. The associated physical grid has 40 latitudes, 48 longitudes, and 18 levels. The standard resolution for the ocean model is a Mercator grid with 128 latitudes, 128 longitudes, and 16 levels. Runs were conducted on three different platforms using configurations of 5 processors (2 by 2 for the atmosphere–land–sea-ice–coupler and 1 for the ocean), 9 processors (2 by 4 plus 1), 17 processors (4 by 4 plus 1), 34 (4 by 8 plus 1 by 2), and 68 processors (4 by 16 plus 2 by 2) processors. (The last configuration comes from atmosphere model requirements for number of local latitudes and number of processors.) Figure 5 shows that the model scales well over the range of configurations tested. The throughput goal for FOAM was chosen to be ten thousand times real time, that is, the ability to simulate 10,000 years in a single calendar year. FOAM meets this goal on a modest number of processors.

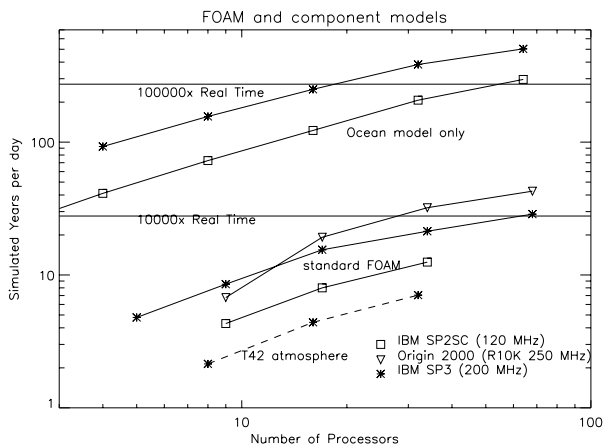


Fig. 5. Timings for FOAM and its components on various platforms for different numbers of processors.

Figure 5 and the processor configurations show how the atmosphere component dominates performance. The ocean model, despite having nearly eight times

as many grid points as the atmosphere, has a much higher throughput when running by itself on the same number of processors. The difference comes from the numerous, detailed physical parameterizations in the atmosphere compared with the ocean and other components. This behavior is seen in other coupled climate models also.

Given the dominance of the atmosphere component on FOAM's performance, the choice of atmosphere resolution is as important for overall throughput as the design of the coupler and the complete model. When the atmosphere model is executed by itself at a higher, more standard resolution of T42 (64 latitudes and 128 longitudes), it is slower by a factor roughly equal to the ratio of the number of points in each grid.

Figure 5 also shows how increases in processor speed have translated into increases in model throughput. Performance nearly doubles between the IBM SP2 and IBM SP3 and nearly doubles again when timed on an IBM SP3 with 350MHz CPU's (not shown).

5 Conclusions

The Fast Ocean Atmosphere Model is a coupled climate model created to address climate science questions that require many simulated years of interaction but not best-possible resolution. Through a combination of a low-resolution parallel atmosphere model, a highly efficient parallel ocean, and a software design that minimizes the effect of coupling on bandwidth and CPU usage, FOAM can simulate several decades of global climate interaction a day on modest numbers of distributed-memory parallel processors. The simulated climate is physically realistic and comparable to models with higher resolution.

The rapid development of FOAM was made possible by the work done by others on the atmosphere and ocean models and also by the simplicity of land and sea-ice components. Work on the next version of FOAM will concentrate on upgrading the representation of the sea-ice and land surface to match that of the NCAR Climate System Model (CSM) [2]. FOAM's current sea ice model lacks many standard features such as ice dynamics and complex thermodynamics. In order to accommodate new models, it may be necessary to relax the conditions FOAM imposes on the land and sea-ice components. Although the resolution of FOAM's ocean model is certainly adequate for modern sea-ice and land models, separately developed components do not usually have the flexibility to change their resolution, grid, or decomposition. A rewrite of FOAM's coupler will be necessary to support arbitrary decompositions of land and sea-ice models; fortunately, that task has been made much easier by the development of new software libraries such as the Model Coupling Toolkit [10].

The relationship between FOAM and larger climate modeling efforts such as CSM is complementary. FOAM uses CSM's submodels wherever possible; adapting them to FOAM's framework and applying them to climate questions that require more throughput than resolution. As large, institutionally supported climate models increase their flexibility and parallelism, it should be

possible to match FOAM's combination of throughput and simulation quality by invoking appropriate options within the larger model. Until then, FOAM development will continue.

Acknowledgments: This work is supported by the Office of Biological and Environmental Research of the U.S. Department of Energy's Office of Science under contracts W-31-109-ENG-38 and DE-FG02-98ER62617.

References

1. Thomas L. Acker, Lawrence E. Buja, James M. Rosinski, and John E. Truesdale. User's Guide to NCAR CCM3. NCAR Tech. Note NCAR/TN-421+IA, Natl. Cent. for Atmos. Res., Boulder, Co., 1996.
2. B. A. Boville and P. R. Gent. The NCAR climate system model, version one. *J. Climate*, 11:1115–1130, 1998.
3. Kirk Bryan. A numerical method for the study of the circulation of the World Ocean. *J. Comp. Phys.*, 4:347–376, 1969.
4. M. D. Cox. A Primitive Equation three-dimensional model of the ocean. Technical Report GFDL Ocean Group Tech. Rep. 1, GFDL, Princeton, NJ, 1984.
5. J. Drake, I. Foster, J. Michalakes, Brian Toonen, and Pat Worley. Design and performance of a scalable parallel community climate model. *Parallel Computing*, 21(10):1571–1591, October 1995.
6. James J. Hack, Byron A. Boville, Bruce P. Briegleb, Jeffrey T. Kiehl, Philip J. Rasch, and David L. Williamson. Description of the NCAR Community Climate Model (CCM2). NCAR Tech. Note NCAR/TN-382+STR, Natl. Cent. for Atmos. Res., Boulder, Co., 1993.
7. Robert Jacob. *Low Frequency Variability in a Simulated Atmosphere Ocean System.* PhD thesis, University of Wisconsin-Madison, 1997.
8. Jeffrey T. Kiehl, James J. Hack, Gordon B. Bonan, Byron A. Boville, Bruce P. Briegleb, David L. Williamson, and Philip J. Rasch. Description of the NCAR Community Climate Model (CCM3). NCAR Tech. Note NCAR/TN-420+STR, Natl. Cent. for Atmos. Res., Boulder, Co., 1996.
9. P. D. Killworth, D. Stainforth, D. J. Webb, and S. M. Patterson. The development of a free-surface Bryan-Cox-Semtner ocean model. *J. Phys. Oceanogr.*, 21:1333–1348, 1991.
10. Jay Larson and Robert Jacob. A message-passing parallel Model Coupling Toolkit (MCT). *in prep.*, 2001.
11. Zhengyu Liu, John Kutzbach, and Lixin Wu. Modeling climate shift of El-Niño variability in the holocene. *Geophy. Res. Lett.*, 27:2265–2268, 2000.
12. James R. Miller, Gary L. Russell, and Guilherme Caliri. Continental-scale river flow in climate models. *J. Climate*, 7:914–928, 1994.
13. Chris Poulsen, Raymond T. Pierrehumbert, and Robert L. Jacob. Impact of ocean dynamics on the simulation of the neoproterozoic "snowball" earth. *Geophy. Res. Lett. in press*, 2001.
14. Michael Tobis. *Effect of Slowed Barotropic Dynamics in Parallel Ocean Climate Models.* PhD thesis, University of Wisconsin-Madison, 1996.
15. David L. Williamson, Jeffrey T. Kiehl, V. Ramanathan, Robert E. Dickinson, and James J. Hack. Description of NCAR Community Climate Model (CCM1). NCAR Tech. Note NCAR/TN-285+STR, Natl. Cent. for Atmos. Res., Boulder, Co., 1987.

The Model Coupling Toolkit

J. Walter Larson[1], Robert L. Jacob[1], Ian Foster[1], and Jing Guo[2]

[1] Argonne National Laboratory, Mathematics and Computer Science Division
9700 S. Cass Ave., Argonne, IL 60439, USA,
larson@mcs.anl.gov
[2] Data Assimilation Office, NASA Goddard Space Flight Center
Greenbelt, MD 20771, USA

Abstract. The advent of coupled earth system models has raised an important question in parallel computing: What is the most effective method for coupling many parallel models to form one high-performance coupled modeling system? We present our solution to this problem—*The Model Coupling Toolkit* (MCT). We describe how our effort to construct the Next-Generation Coupler for NCAR Community Climate System Model motivated us to create the Toolkit. We describe in detail the conceptual design of the MCT, and explain its usage in constructing parallel coupled models. We present some preliminary performance results for the Toolkit's parallel data transfer facilities. Finally, we outline an agenda for future development of the MCT.

1 Introduction

In recent years, climate modeling has evolved from the use of atmospheric general circulation models (GCMs) to *coupled* earth system models. These coupled models comprise an atmospheric GCM, an ocean GCM, a dynamic-thermodynamic sea ice model, a land-surface model, and a *flux coupler* that coordinates data transfer between the other component models, and governs the overall execution of the coupled model (Figure 1). Coupled models present a considerable increase in terms of computational and software complexity over their atmospheric model counterparts.

The flux coupler typically serves the following functions: overall command and control of the coupled model, including syncronization, error/exception handling, and intitialization and shutdown of the system; communication of data between component models; time averaging and accumulation of data from one component for use in subsequent transmission of data to other components; computation of interfacial fluxes for component given state data from other components; interpolation of flux and state data between the various component model grids; merging of flux and state data from multiple components for delivery to yet another component.

V.N. Alexandrov et al. (Eds.): ICCS 2001, LNCS 2073, pp. 185–194, 2001.
© Springer-Verlag Berlin Heidelberg 2001

The computational demands of each of the component models are sufficient to require message-passing parallelism (and in some cases hybrid parallelism) in each component model to achieve high performance on microprocessor-based distributed memory computers. This creates a formidable challenge in coupling these models together. This challenge manifests itself in a potentially high degree of computational and software complexity in the flux coupler. The coupler must be aware of all the component models, the grids on which they present and require data, and their repsective data decompositions. The coupler must be able to handle all of this information, and serve the information required by the component models in a timely fashion, lest it cause the coupled system to hang.

Various coupled model and coupler architectures have been created in the attempt to meet the aforementioned challenges. A general diagram for coupled model architecture is shown in Figure 2, which shows the main parts of a coupled model with four component models as a "wheel": the syncrhonization and command/control apparatus (the "rim" of the wheel); the coupler computatonal core (the "hub" and "spokes" of the wheel); the component models (*e.g.*, atmosphere, ocean, sea-ice, and land); the component model-coupler interfaces.

There are five main architectural approaches for coupled models:

1. A single-executable "event loop" coupled model, in which a single pool of processers us used for the coupled model, with each the component models and the coupler running in turn. Execution of the model under this architecture can be viewed as a sweep second hand revolving around the wheel diagram in Figure 2. An example of this architecture is the Parallel Climate Model (PCM).
2. A single-executable asynchronous coupled model, with each of the component models and coupler runnin simultaneously and exchanging data as needed.
3. A multiple-executable asynchronous coupled model, with each of the component models and coupler runnin simultaneously and exchanging data as needed. An example of this architecture is the current version of the NCAR CCSM.
4. A single-executable asynchrouns model in which the functions of the flux coupler are distributed among the various component models. In terms of Figure 2, the flux coupler (the "hub" and the "spokes" of the wheel) disappear. An example of this coupling strategy is the Fast Ocean-Atmosphere Model (FOAM).
5. A multiple-executable asynchrouns model in which the functions of the flux coupler are distributed among the various component models.

We took as our primary coupler design requirement the ability to support each of the five aforementioned coupled model architectures.

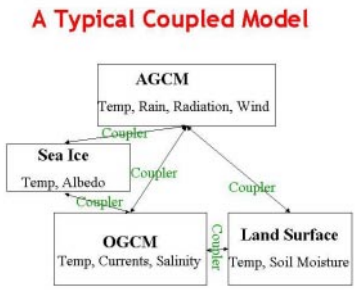


Fig. 1. Schematic of a coupled earth system model, showing the component models and examples of the flux and state data they exchange.

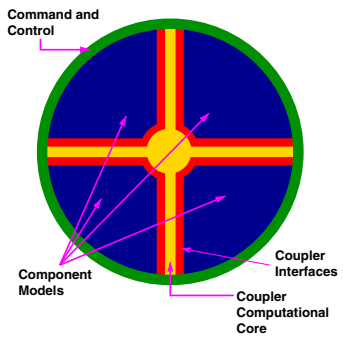


Fig. 2. Schematic of a coupled modeling system, with component models.

2 The Community Climate System Model Next-Generation Coupler

The problem that motivated the creation of the model coupling toolkit was the Accelerated Climate Prediction Initiative (ACPI) *Avant Garde* project, whose goal is the creation of a modular, performance- portable Community Climate System Model (CCSM). One major task in this project is the design and implementation of a modular, extensible, high-performance "Next-Generation Coupler" (NGC) for the CCSM.

A full statement of the requirements for the NGC are given at the URL

`http://www.cgd.ucar.edu/csm/models/cpl-ng/#requirements`

Two types of requirements were identified—*scientific requirements* and *computational functionality requirements*. The scientific requirements outline the coupler's role in the coupled modeling system, and a list of the core functions the coupler must provide. The computational functionality requirements outline the programming language(s) to which the coupler must provide interfaces, and portability and performance issues.

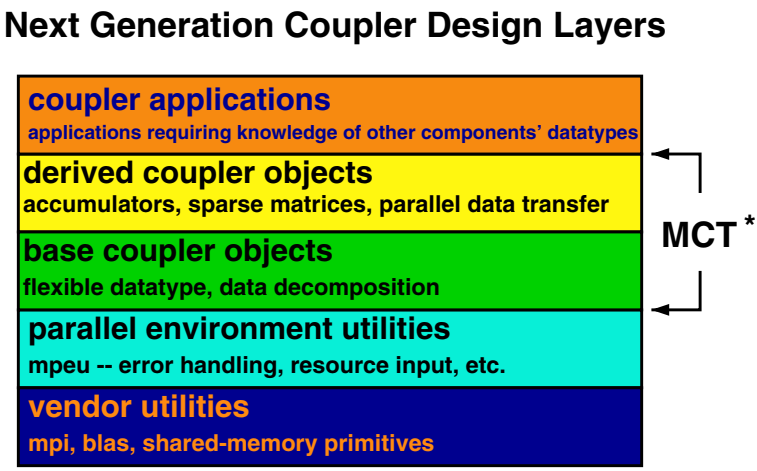


Fig. 3. Layered design for the CCSM Next-Generation Coupler.

Analysis of the two groups of requirements yielded a layered design. The Layers (Figure 3), ranked lowest-level to highest-level are: Vendor Utilities, which include standard libraries, vendor implementations of the Message-Passing Interface (MPI) library, and shared-memory primitives (*e.g.*, SHMEM); Parallel Environment Utilities, which include the Data Assimilation Office's Message Passing Environment Utilities (mpeu) library, and LBL's Message-Passing Handshaking (MPH) utilities; Basic Coupler Classes and Methods, which include low-level MCT objects such as the internal data representation and data decomposition descriptors; Derived Coupler Classes and Methods, which include the MCT datatypes and routines to support: interpolation implemented as sparse matrix-vector multiplication; time averaging; computation of fluxes from state variables; Coupler Applications, which are the coupler computational core and component model-coupler interfaces built using MCT components, and facilities for converting component model datatypes and domain decomposition descriptors into MCT components.

3 The Model Coupling Toolkit

The model coupling toolkit is a set of software components that ease the programming of parallel coupled modeling systems. The main services provided by the toolkit are: data decomposition descriptors; a flexible, extensible, indexible field storage datatype; support for time averaging and accumulation; data field interpolation implemented as sparse matrix-vector multiplication; intercomponent communications and parallel data transfer.

3.1 Description and Underlying Assumptions

The MCT is highly modular, and implemented in Fortran 90. All the toolkit functions feature explicit `INTENT` declarations in their interfaces and extensive argument and error checking. Toolkit modules and routines have prologues that can be processeed using the tool ProTeX to create LaTeX documentation. Further information regarding the toolkit, including complete documentation is available at the MCT web site:

`http://www.mcs.anl.gov/~larson/mct`

The MCT relies on the following underlying assumptions: each component has its own MPI communicator; each component has a unique integer ID; each component is on a distinct set of processors; interpolation implemented as sparse matrix-vector multiplication; components can exchange only real and integer data as groups of vectors. The MCT user must supply a consistent numbering schemes for grid points for each component model grid, and the interpolation matrix elements. Once the user has satisfied these assumptions and requirements, the MCT allows the user to link any number of component models, using any grid, any domain decomposition, and any number of processors-per component model.

3.2 Toolkit Components

The low-level components in the MCT are the internal data field representation and data decomposition descriptors. The high-level components in the toolkit are: a component model registry; time-averaging and accumulation registers; sparse matrix storage; the grid geometry description; the communications scheduler. The high and low-level components are presented in Table 3.2.

Fields are represented in the MCT in its own internal data structure called an *attribute vector*, and is implemented in the `AttrVect` component. The `AttrVect` is used extensively in the Toolkit's interpolation, time-averaging, and parallel data transfer functions. The `AttrVect` component is defined as a Fortran 90 derived type:

```
Type AttrVect
  type(List) :: iList
  type(List) :: rList
  integer, dimension(:,:), pointer :: iAttr
  real, dimension(:,:), pointer :: rAttr
End Type AttrVect
```

The `List` components `iList` and `rList` list the integer and real attributes (fields) of the `AttrVect` respectively. A `List` is a string, with substrings delimited by colons. Suppose we wish to store the real fields for surface zonal and meridonal winds and temperature in an `AttrVect` component. We first define

string tags for each field: `us` for surface zonal wind; `vs` for surface meridional wind; `ts` for surface temperature. For this example, the `rList` component would be rList = 'us:vs:ts'.

These fields can be accessed by using an `AttrVect` inquiry, and supplying the string tag to reference the desired field.

The `AttrVect` is a fundamental data type in the toolkit, and is used for other purposes than field storage. Other uses include: time averaging and accumulation registers in the `Accumulator` component; sparse matrix element storage in the `SparseMatrix` component; grid point coordinate and area/volume weight storage in the `GeneralGrid` component.

The MCT has two basic types of domain decomposition descriptors: the `GlobalMap`, which describes a simple, one-dimensional data decomposition, and the `GlobalSegMap`, which describes a segmented data decomposition capable of describing multidimensional decompositions of multidimensional grids and unstructured grids. The `GlobalMap` type is a special, simple case of the more general `GlobalSegMap`. Users of the MCT must translate their domain decompositions into either the `GlobalMap` or `GlobalSegMap` form.

A simple example of how the decomposition of a two dimensional grid is stored in a `GlobalSegMap` is shown in Figure 4.

The MCT provides numerous facilities for manipulating and exchanging the `GlobalMap` and `GlobalSegMap` domain descriptor components, including: inter-component exchanges of maps; global-to-local index translation; local-to-global index translation.

A full description of the higher-level components of the MCT are beyond the scope of this paper, and we have summarized the components and their methods (excluding the create and destroy methods) are summarized in Table `tab:mct`.

The one high-level portion of the MCT we will describe here is the parallel data transfer. Parallel data transfer is accomplished by creating a communications scheduler called a `Router` from the domain descriptors of the source and

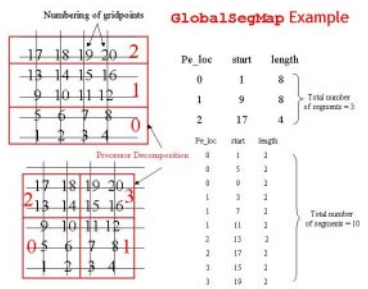


Fig. 4. Illustration of the `GlobalSegMap` domain decomposition descriptor component.

target component models. An example of how a `Router` is created is shown in Figure 5. Once the appropriate `Router` is created, the parallel transfer is effected by calling the routines `MCT_Send` and `MCT_Recv()` on the source and target component models, respectively. The only arguments to these transfer routines are: a `Router` to coordinate the parallel send (receive); an `AttrVect` to store the input (ouptut) data.

Table 1. Components of the Model Coupling Toolkit

Service	Component	Methods
Data Storage	`AttrVect`	Indexing Sorting Gather, Scatter, Broadcast
Domain Decomposition	`GlobalMap` `GlobalSegMap`	Indexing Exchange
Time Average/Accumulation	`Accumulator`	Accumulate methods from `AttrVect`
Interpolation	`SparseMatrix`	Multiplication methods from `AttrVect`
Grid Description	`GeneralGrid`	area/volume integrals methods from `AttrVect`
Component Registry	`MCTWorld`	component identificaiton process address translation (*i.e.*, from local communicator to `MPI_COMM_WORLD`
Communications Scheduling	`Router`	Parallel transfer routines `MCT_Send()` and `MCT_Recv()`

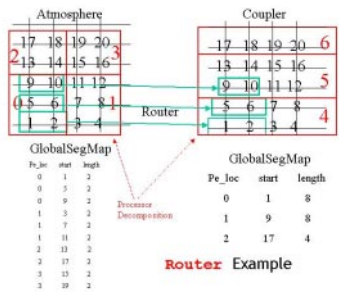


Fig. 5. Illustration of the `Router` communications scheduler component.

4 Usage

Programming of flux couplers and component model-coupler interfaces is accomplished by directly invoking components of the toolkit. A complete example of

how the MCT is used to create a coupled model is beyond the scope of this article. Instead, we shall focus on the MCT unit tester, which implements the simple example of an atmosphere coupled to and ocean via a flux coupler (Figure 6).

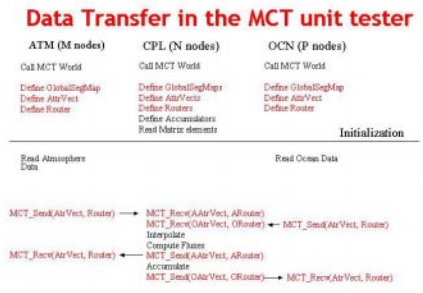


Fig. 6. A simple Atmosphere-Ocean Coupled Model built using the MCT.

5 Performance

Performance characterization of the toolkit has just begun, but some preliminary results concerning the most crucial component–the parallel data transfer routines `MCT_Send` and `MCT_Recv`–are available. Performance of the parallel transfer is highly sensitive to a number of variables, including the number of MPI processes in the sending and receiving component models, the complexity of the source and target as measured by the numbers of segments in their respective `GlobalSegMap` descriptors, and the complexity of the interrelationship and overlaps between the source and target domain decompositions. We present prformance results for the transfer of sixteen T42 (128 longitudes by 64 latitudes) atmospheric grid fields between the atmosphere and flux coupler. We present results for two examples that are meant to capture the extremes of the governing performance factors cited above: (1) a simple example in which the number of MPI processes on the atmosphere and coupler are identical, as are their domain decompositions of the atmosphere grid.; and (2) a complicated example in which the atmosphere has many more MPI processes than the coupler and the atmosphere and coupler domain decompositions are not related in any simple fashion.

Case (1) has atmosphere and coupler decompositions as shown in the left and center panels of Figure 7. The performance of `MCT_Send` and `MCT_Recv`, as measured on an IBM SP3 (375 MHz), is shown in the right panel of Figure 7. The performance for this simple case is as expected: transfer time decreases as the message size decreases and the number of processors assigned to each model is increased.

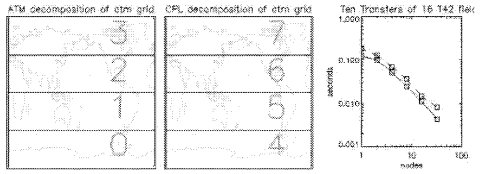


Fig. 7. Atmosphere and Coupler component models with identical domain decompositions.

The domain decompositions for case (2) are shown in the left and center panels of Figure 8. The Router between these two decompositions was automatically determined by MCT. Timing data are shown in right panel of Figure 8. The number of coupler nodes was varied for each of three cases: with the atmosphere on 8 (black), 16 (red), and 32 (blue) nodes. The poor scaling may be an unavoidable result of doing a parallel data transfer between two very dissimilar decompositions. Still, the overall transfer time is very small compared with the time the full model will spend computing 10 timesteps; moreover, the user/developer is relieved of the burden of determining the complex data transfer pattern.

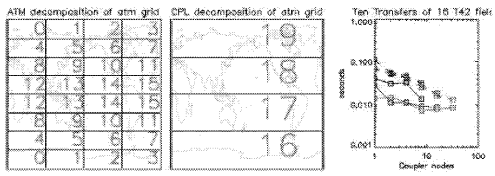


Fig. 8. Atmosphere and Coupler component models with differing numbers of processes and domain decompositions.

Future versions of the toolkit will offer explicit support for dynamic load balancing of component models, assuming the number of MPI processes per component model is held fixed. Accomodating this feature will require the study and optimization of a number of toolkit component methods including: the initialization method for the `Router` component; initialization methods for the `GlobalMap` and `GlobalSegMap` components; domain decomposition descriptor exchange methods.

6 Conclusions and Future Work

A modular Model Coupling Toolkit has been described, and its usage and performance have been discussed. The future development agenda of the MCT includes numerous enhancements: support for on-line interpolation matrix element generation; performance optimization and inclusion of OpenMP to implement hybrid parallelism; upwards abstraction of data types to greater simplify the construction of flux couplers and component model-coupler interfaces; support for higher-dimensional data storage classes; support for higher-dimensional data decomposition descriptor classes; extension to support dynamically load balanced component models (but with fixed process pool sizes); and extension to support dynamically load balanced component models (with dynamically varying process pool sizes).

Acknowledgements: The authors wish to thank many people for the informative and inspiring discussions that helped guide this work: Tom Bettge, Tony Craig, Cecelia Deluca, Brian Kaufman, and Mariana Vertenstein of the National Center for Atmospheric Research; John Michalakes and John Taylor of the Mathematics and Computer Science Division of Argonne National Laboratory. This work is part of the Accelerated Climate Prediction *Avant Garde* project, and is supported by the Office of Biological and Environmental Research of the U.S. Department of Energy's Office of Science.

References

[Jones, 1999] Jones, P. W. (1999). First- and Second-order Conservative Remapping Schemes for Grids in Spherical Coordinates. *Monthly Weather Reveiw*, 127:2204–2210.

Parallelization of a Subgrid Orographic Precipitation Scheme in an MM5-Based Regional Climate Model

L. Ruby Leung[1], John G. Michalakes[2], and Xindi Bian[1]

[1] Atmospheric Science and Global Change Resource, Pacific Northwest National Laboratory, Richland, WA 99352, USA
{ruby.leung, xindi.bian}@pnnl.gov
[2] Mathematics and Computer Science Division, Argonne National Laboratory, Argonne, IL 60439, USA
michalak@mcs.anl.gov

Abstract. Regional Climate Models (RCMs) are practical downscaling tools to yield regional climate information for assessing the impacts of climate variability and change. The Pacific Northwest National Laboratory (PNNL) RCM, based on the Penn State/NCAR Mesoscale Model (MM5), features a novel subgrid treatment of orographic precipitation for coupling climate, hydrologic, and ecologic processes at the watershed scale. The parameterization aggregates subgrid variations of surface topography into a finite number of surface elevation bands. An airflow model and a thermodynamic model are used to parameterize the orographic uplift/descent as air parcels cross over mountain barriers or valleys. The parameterization has significant performance advantages over nesting to achieve comparable resolution of climate information; however, previous implementations of the subgrid scheme required significant modification to the host MM5 model, prohibiting its incorporation within the NCAR-supported community version of MM5. With this effort, software engineering challenges have been addressed to incorporate, parallelize, and load-balance the PNNL subgrid scheme with minimum changes to MM5. The result is an efficient, maintainable tool for regional climate simulation and a step forward in the development of an MM5-based community regional climate model.

1 Introduction

In areas with heterogeneous surface elevation and vegetation, high spatial resolution is required to accurately simulate precipitation and surface hydrology. Since computational cost increases approximately as the cube of resolution in atmospheric models, techniques such as nesting are often used to focus costly high-resolution computation where it is needed. However, the use of nesting to resolve topography in climate simulations has a number of disadvantages. First, topography may be highly spatially variable from cell to cell in a model domain so that even within the limited area of a high-resolution nest, computation is wasted in areas of the nested domain that do not require it. Second, major climate processes that require additional topographical refinement involve

V.N. Alexandrov et al. (Eds.): ICCS 2001, LNCS 2073, pp. 195–203, 2001.
© Springer-Verlag Berlin Heidelberg 2001

only column physics (e.g., cloud and precipitation processes), not model dynamics; therefore, increased temporal resolution—that is, reducing the time step—is unnecessary. A subgrid parameterization of orographic precipitation [4,3,5] has been developed as an alternative to the use of nesting to efficiently perform long-term integration that yields high spatial resolution climate information, which is important for hydrological applications and climate impact assessments.

The subgrid parameterization was first implemented in the prototype version of the Penn State/NCAR Mesoscale Model (MM5) [1] model, producing the PNNL Regional Climate Model, a separate version. Subsequently, the community mesoscale model, now MM5 version 3, underwent additional development including nonhydrostatic dynamics, new options relating to land surface processes, and adaptation to distributed-memory scalable computing systems [6]. The demonstrated effectiveness of the same-source parallelization approach and the need for scalable performance in the PNNL RCM suggested the feasibility and appropriateness of integrating the PNNL-developed climate parameterizations into the NCAR-supported community version of the model. Ultimately, this effort is intended to form the basis for an MM5-based Community Regional Climate Model (CRCM). To date, we have developed a new parallel version of the PNNL RCM and subgrid scheme based on the current (at this writing) MM5 Version 3.4. Section 2 of this paper describes the host MM5 model. Section 3 describes subgrid scheme and details of parallelization and load balancing. Section 4 provides preliminary performance and load balancing results. Section 5 summarizes the issues addressed in this community effort.

2 Penn State/NCAR MM5

MM5 is a limited-area, nonhydrostatic, terrain-following sigma-coordinate model designed to simulate mesoscale and regional-scale atmospheric circulation. Features of the models include (i) a multiple-nest capability, (ii) nonhydrostatic dynamics, which allows the model to be used at a few-kilometer scale, (iii) multitasking capability on shared- and distributed-memory machines, (iv) a four-dimensional data-assimilation capability, and (v) numerous physics options. The latest versions of MM5 include several features that are important for climate applications such as regular updating of the lower boundary conditions (sea surface temperature and sea ice), two options for land surface modeling, and physics options such as radiative transfer that are more accurate for long-term integrations.

Parallelism in MM5 is implemented by using the "same-source" approach, described in [6]. This involves a traditional two-dimensional horizontal data domain decomposition, but with minimal changes to the original source, allowing the parallel code to be maintained as part of the official MM5 for use on a range of both shared- and distributed-memory parallel computers, including the IBM SP, Cray T3E, Fujitsu VPP, Compaq ES40, SGI Origin2000, PC and Alpha-based Beowulf clusters, and workstations. Interprocessor communication to update subdomain halos, exchange forcing and feedback data between nested

domains, and implement distributed I/O is supplied by the RSL library [7]. RSL also provides support for automatic domain decomposition with unequally sized subdomains for load balancing. The FLIC source translator automates changes to MM5 loops and indices for parallel computation [8]. More recently, FLIC has been extended to collapse MM5 physics loops over the two horizontal dimensions into a single loop for improved performance on vector machines, and this translation is identical to that required for the PNNL subgrid scheme. More details are provided in the following section.

3 Subgrid Orographic Precipitation Scheme

The subgrid parameterization of orographic precipitation aggregates subgrid variations of surface topography into a finite number of surface elevation bands. A dominant vegetation cover is defined for each elevation class of each model grid cell to account for the subgrid heterogeneity in vegetation and land-water contrast. An airflow model and a thermodynamic model are used to parameterize the orographic uplift/descent as air parcels cross over mountain barriers or valleys. Physical processes such as cloud microphysics, convection, turbulence transfer, radiative transfer, and land-atmosphere transfer are all calculated for each subgrid elevation class based on its surface elevation, vegetation cover, and atmospheric conditions. The result is separate predictions of precipitation, temperature, snow water equivalent, soil moisture, and surface runoff for a selected number of surface elevation classes within each grid cell.

Figure 1 shows a schematic of the subgrid parameterization as applied to a grid cell 50 km by 50 km in the western United States. During postprocessing, the simulated fields can be distributed according to the spatial distribution of surface elevation within each grid cell to yield predictions at the scale of the surface elevation data. Hence, the RCM can operate at a coarser spatial resolution (typically 50–100 km) while still accounting for subgrid spatial heterogeneity in surface topography and vegetation, but at a reduced computational cost. The subgrid method significantly improves the simulation of surface temperature, precipitation, and snowpack over mountainous areas (see, e.g., [2]).

3.1 Structure and Decomposition of Subgrid Variables

When running with the subgrid parameterization of orographic precipitation, MM5 computes two solutions of the physically forced prognostic equations, one for the grid cell mean variables, and one for the subgrid variables. A separate set of arrays stores the subgrid variables. These include prognostic variables (e.g., temperature, wind, moisture, the various forms of cloud water, and the pressure perturbation) and their tendencies, and diagnostic variables such as precipitation and ground temperature. Several new arrays are added to store information for mapping between the grid cells and subgrid elevation classes.

SOLVE is the MM5 subroutine that computes the main physics and dynamics at each model time step. It includes calls to all advection, diffusion, time-split

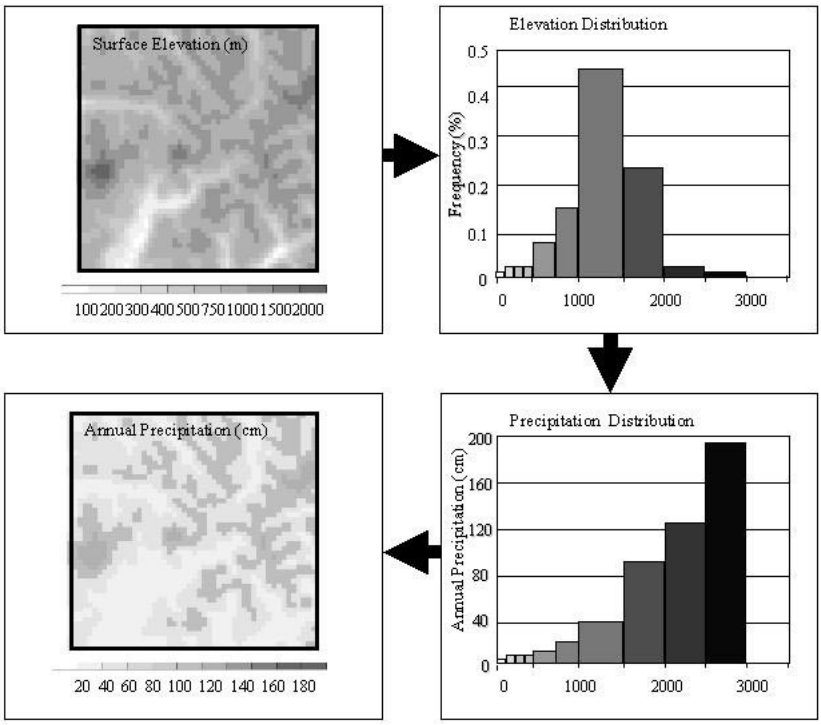


Fig. 1. Schematic illustration of the subgrid parameterization of orographic precipitation applied to a grid cell 50 km by 50 km in the western U.S. Upper left: surface topography within the grid cell at 1 km spatial resolution. Upper right: subgrid elevation classification. Lower right: simulations of precipitation at each subgrid elevation class. Lower left: mapping of precipitation to the geographical area based on elevation to yield high spatial resolution distribution of climate conditions for driving hydrology models

integration, and model physics routines. Most of these routines are called from SOLVE within loops over the J (east-west) dimension and compute one sweep of the I (north-south) dimension each time they are called. With the subgrid scheme, the enclosing J loops in the SOLVE routine are removed. Within the subroutines, the horizontal indices (I,J) are collapsed so that the iteration sweeps over the single index that runs from 1 to NHT, where NHT is the total number of subgrid elevation classes of all grid cells. This collapsing of indices is identical to a translation that FLIC performs for performance improvements on vector machines. Thus, the approach to integrating the PNNL subgrid parameterization easily leverages the overall same-source infrastructure already employed in MM5.

Parallelization involves decomposing subgrid arrays so that subgrid classes are on the same processor as the corresponding cells in the regular model grid. The NHT_{global}-sized elevation class arrays are decomposed so that the NHT_{local}-sized arrays on each processor contain variables for the elevation

classes corresponding to the grid cells in the local processor's subdomain. Because the distribution of elevation classes over grid cells is nonuniform, NHT_{local} may vary considerably from processor to processor in a simple equal-area decomposition. This is the basic source of load imbalance associated with the subgrid scheme.

3.2 Load Balancing

Load imbalance results from unequal distribution of subgrid elevation classes (NHT_{local}) when a domain is decomposed over processors. This load imbalance is static because NHT depends only on the spatial heterogeneity in surface topography that is determined once the domain is selected. Since a load-balancing mechanism already exists in the parallel MM5, a simple and effective approach to balancing the number of elevation classes is to redistribute the grid cells with which these classes are associated. We redistribute the cells in the regular MM5 grid to maximize $\frac{NHT_{global}}{p \ max_p(NHT_{local})}$, where p is the number of processors and max_p is the maximum over p processors. A load-balanced decomposition is computed at the beginning and remains in force for the duration of the model run.

The algorithm used to compute the decomposition is only a slight modification of the MM5 algorithm, which weights grid cells according to whether they are interior domain points (higher computational cost associated with physics calculations) or boundary points and then computes a decomposition that yields subdomains having close to the same aggregate weights. The subgrid load-balancing algorithm includes an additional cell-weighting factor, called *band-influence,* which determines the influence of the number of elevation classes associated with a grid cell. Figure 2a shows the static imbalance associated with the subgrid scheme in a 70 by 70 cells domain covering the western United States. The subgrid parameterization is applied to the interior 50 by 40 grid cells. The distribution of elevation bands is determined by a terrain dataset at 1 km spatial resolution. Cells with higher numbers of elevation classes are shown in lighter colors; black represents cells with only a single elevation class. The latter are found mostly over the ocean and near the boundaries where the subgrid scheme is not applied.

Figure 2b shows a 64-processor decomposition computed with a band-influence of zero—grid cells distributed more or less equally to different processors. The cost for an average time step on this domain is 513 milliseconds on an IBM SP. Figure 2c shows the domain decomposition that is computed with band-influence of one. The sizes of the local processor subdomains are varied to achieve a more uniform distribution of subgrid elevation bands to each processor. Here, the cost for an average time step is 269 milliseconds, a significant improvement.

4 Performance Results

A series of runs was performed on the IBM SP system at NCAR (Winterhawk-II, 4 375 MHz processors per node) using a 50 km resolution domain covering

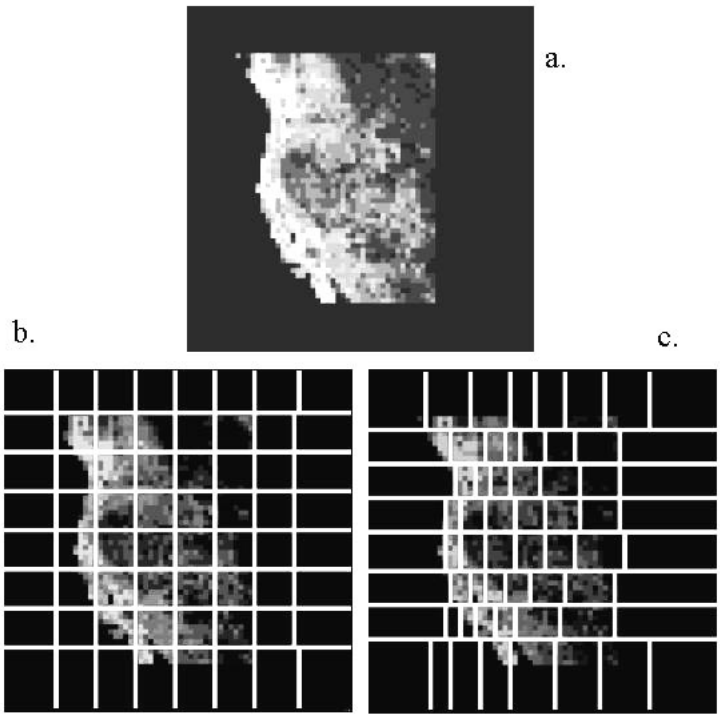


Fig. 2. Distribution of subgrid elevation bands in the 70 by 70 grid cells domain in the western U.S. (a) lighter cells have more elevation bands (max. 12); black is single band per cell. Multiple elevation bands per cell create static load imbalance; (b) simple decomposition (band-influence=0.0); and (c) load-balanced decomposition (band-influence=1.0)

the western United States. The domain, as shown in Figure 2a, consists of 70 by 70 cells in the horizontal with 23 vertical layers. The time step was 150 seconds. Runs were conducted on 16, 36, 64, and 100 processors (4x4, 6x6, 8x8, and 10x10 decompositions, respectively). All runs were straight MPI, with four MPI processes per node. In each set of runs, the band-influence parameter in the modified MM5 load-balancing algorithm was varied from 0.0 (no influence of subgrid imbalance) to 1.1. Performance of each time step was measured using a millisecond timer and averaged over the last half-hour of a three-hour simulation, allowing sufficient time for spin-up of moisture fields. Initialization and I/O cost were ignored.

Figure 3 shows performance for these runs expressed as the number of model time steps executed per wall-clock second. All four sets showed improvement as band-influence was increased. Band-influence=1.0 was optimal for the 16, 36, and 64 processor runs; band-influence=0.9 was optimal for the 100 processor run. Load balancing provided a 96 percent improvement over the nonload-balanced

performance on 100 processors, 91 percent improvement on 64 processors, 88 percent on 36 processors, and 71 percent on 16 processors. The results indicate that for the processor counts tested, the benefit of load balancing increases with the number of processors. One expects this situation eventually to reverse with higher numbers of processors, however, since smaller subdomains will provide less opportunity for load balancing by redistributing grid points.

As expected, load balancing improves scaling efficiency, the speedup divided by the increase in the number of processors. Scaling with load balancing is 87 percent from 16 to 36 processors, 79 percent from 16 to 64 processors, and 64 percent from 16 to 100 processors. Scaling without load balancing is 79 percent from 16 to 36 processors, 70 percent from 16 to 64 processors, and 56 percent from 16 to 100 processors.

Performance of the subgrid scheme for the tested example compares quite favorably with traditional nesting. Employing the load-balanced subgrid scheme over half the area of the total domain required 2.05 times longer to run on 16 processors, 2.04 times on 36 processors, 1.7 times on 64 processors, and 1.98 times on 100 processors than without the subgrid scheme. A similarly sized nest would cost 14.5 times more than without a nest: an additional 4.5 times the number of cells time-stepping three times more frequently, plus the time for the original coarse domain.

5 Conclusions

Regional climate models are downscaling tools that enable the understanding and predictions of regional response to large-scale climate forcings. They can be used to provide spatially detailed seasonal climate forecasts and long-term climate projections useful for managing natural resources as well as serving as testbeds for developing physics parameterizations for global climate models.

Although the Penn State/NCAR Mesoscale Model MM5 was originally developed for short-term simulations of mesoscale weather phenomena, a community effort has been organized to add a capability for regional climate simulations. The following issues are being addressed to develop a Community Regional Climate Model (CRCM) based on MM5: (i) computational efficiency, (ii) stable numerics for long-term integration, (iii) lateral boundary condition formulation, (iv) a suite of physical parameterizations that provide accuracy and computational efficiency for long-term simulations, (v) model pre- and postprocessing, and (vi) a well-coordinated testing of different model components to ensure its suitability for long-term integration. As part of this community effort, the subgrid parameterization of orographic precipitation developed by Leung and Ghan has been implemented in the parallel MM5. This parameterization provides a computationally efficient alternative to the use of nesting for achieving simulations with high spatial resolution. With the example illustrated, model execution time increased approximately 2-fold using the subgrid scheme; the cost increase for a comparable refinement using a nested domain would have been 14-fold.

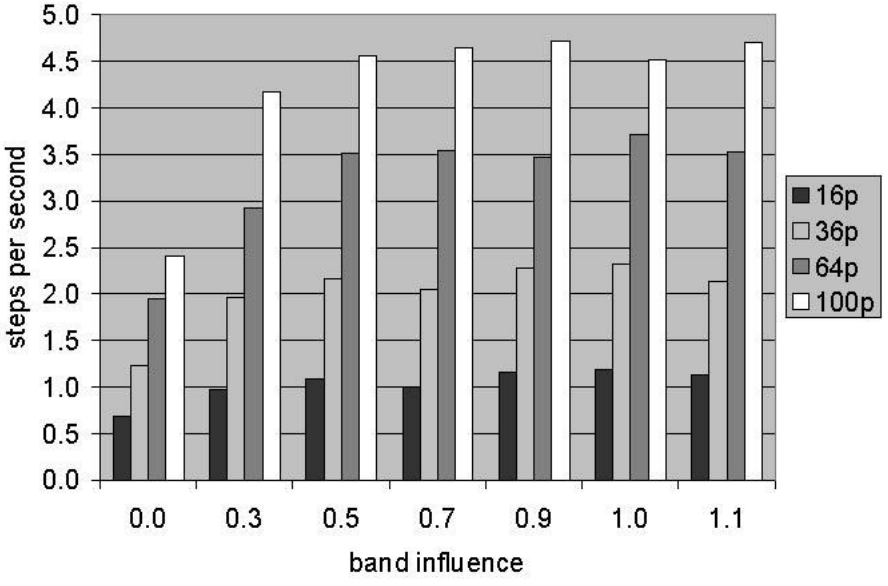


Fig. 3. Performance of a 70 by 70 cell domain on 20, 36, 64, and 100 IBM SP processors, varying the band-influence parameter in the MM5 load-balancing algorithm. Band-influence=0.0 disregards the load imbalance associated with the subgrid scheme; band-influence=1.0 gives each elevation class the full weight of an additional grid cell on the processor

Thus, the parallelized subgrid parameterization with load balancing represents a 7-fold savings over traditional nesting in MM5 for this scenario.

The implementation of the subgrid parameterization is consistent with the same-source approach to parallelization and vectorization adopted by the standard MM5. Changes are mostly transparent to users who opt not to use the parameterization. Load balancing is an important issue when the parameterization is applied to spatially diverse regions where the number of subgrid elevation classes vary strongly from one grid cell to another. The load-balancing algorithm in MM5 was modified to address this issue. In the near future, this parameterization will be implemented in the Weather Research and Forecast Model to improve its capability for regional climate simulations. Application of the subgrid scheme to enhance regional resolution in global climate simulations is also under way. The scheme has been implemented experimentally in the NCAR CCM3, and evaluation is being performed over different regions of the world.

References

1. Grell, G. A., J. Dudhia, and D. R. Stauffer: A Description of the Fifth-Generation Penn State/NCAR Mesoscale Model (MM5). Tech. Rep. NCAR/TN-398+STR, National Center for Atmospheric Research, Boulder, Colorado (1994).
2. Leung, L. R., and S. J. Ghan: Pacific Northwest Climate Sensitivity Simulated by a Regional Climate Model Driven by a GCM. Part I: Control Simulation. J. Clim. **12** (1999) 2010–2030.
3. Leung, L. R., and S. J. Ghan: Parameterizing Subgrid Orographic Precipitation and Surface Cover in Climate Models. Mon. Wea. Rev. **126** (1998) 3271–3291.
4. Leung, L. R., and S. J. Ghan: A Subgrid Parameterization of Orographic Precipitation. Theor. Appl. Climatol. **52** (1995) 2697–2717.
5. Leung, L. R., M. S. Wigmosta, S. J. Ghan, D. J. Epstein, and L. W. Vail: Application of a Subgrid Orographic Precipitation/Surface Hydrology Scheme to a Mountain Watershed. J. Geophys. Res. **101** (1996) 12803–12817.
6. Michalakes, J.: The Same-Source Parallel MM5. J. Sci. Programming **8** 2000 5–12.
7. Michalakes, J.: RSL: A Parallel Runtime System Library for Regional Atmospheric Models with Nesting, in *Structured Adaptive Mesh Refinement (SAMR) Grid Methods,* IMA Volumes in Mathematics and Its Applications (117), Springer, New York, 2000, pp. 59–74.
8. Michalakes, J.: FLIC: A Translator for Same-source Parallel Implementation of Regular Grid Applications, Tech. Rep. ANL/MCS-TM-223, Argonne National Laboratory (1997).

Resolution Dependence in Modeling Extreme Weather Events

John Taylor,[1,2] *Jay Larson*[1]

[1]Mathematics & Computer Science, Argonne National Laboratory, Argonne, Illinois 60439
[2]Environmental Research Division, Argonne National Laboratory, Argonne, Illinois 60439
{jtaylor larson}@mcs.anl.gov
http://www-climate.mcs.anl.gov/

ABSTRACT. At Argonne National Laboratory we have developed a high performance regional climate modeling simulation capability based on the NCAR MM5v3.4. The regional climate simulation system at Argonne currently includes a Java-based interface to allow rapid selection and generation of initial and boundary conditions, a high-performance version of MM5v3.4 modified for long climate simulations on our 512-processor Beowulf cluster (Chiba City), an interactive Web-based analysis tool to facilitate analysis and collaboration via the Web, and an enhanced version of the CAVE5d software capable of working with large climate data sets. In this paper we describe the application of this modeling system to investigate the role of model resolution in predicting extreme events such as the "Hurricane Huron" event of 11-15 September 1996. We have performed a series of "Hurricane Huron" experiments at 80, 40, 20, and 10 km grid resolution over an identical spatiotemporal domain. We conclude that increasing model resolution leads to dramatic changes in the vertical structure of the simulated atmosphere producing significantly different representations of rainfall and other parameters critical to the assessment of impacts of climate change.

1 Introduction

In a recent IPCC report on *The Regional Impacts of Climate Change* it was concluded:

The technological capacity to adapt to climate change is likely to be readily available in North America, but its application will be realized only if the necessary information is available (sufficiently far in advance in relation to the planning horizons and lifetimes of investments) and the institutional and financial capacity to manage change exists. [1]

IPCC also acknowledged that one of the key uncertainties that limit our ability to understand the vulnerability of subregions of North America to climate change and to develop and implement adaptive strategies to reduce vulnerability was the need to develop accurate regional projections of climate change, including extreme events [1]. In particular, we need to account for the physical-geographic characteristics that play a significant role in the North American climate (e.g., the Great Lakes, coastlines, and mountain ranges) and also properly account for the feedbacks between the biosphere and atmosphere [1].

V.N. Alexandrov et al. (Eds.): ICCS 2001, LNCS 2073, pp. 204–211, 2001.
Springer-Verlag Berlin Heidelberg 2001

The potential impacts of global climate change have long been investigated based on the results of climate simulations using global climate models with typical model resolutions of the order of hundreds of kilometers [2,3]. However, the assessment of the impacts of climate change at the regional and local scales requires predictions of climate change at the 1-10 kilometer scale. Model predictions from global climate models with such high resolutions are not likely to become widely available in the near future.

Accordingly, at Argonne National Laboratory we have begun developing a regional climate simulation capability for high-performance computers with the long-term goal of linking the predictive global climate modeling capability with the impact assessment and policymaking communities. The primary technical challenge is to downscale global climate model output to the regional scale. Our focus area is the Midwest region of the United States.

2 Argonne Regional Climate Simulation System

The regional climate simulation system at Argonne currently includes

- a Java-based interface to allow rapid selection and generation of initial and boundary conditions,
- a high-performance version of MM5v3.4 modified to enable long climate simulations on the Argonne Chiba City 512-processor (500 MHz Pentium III) Beowulf cluster,
- an interactive Web-based analysis tool to facilitate analysis and collaboration via the Web, and;
- an enhanced version of the CAVE5d software capable of working with large climate data sets.

The model used in this study is the Pennsylvania State University/National Center for Atmospheric Research (PSU/NCAR) fifth-generation mesoscale model (MM5). In brief, MM5 is a three-dimensional, nonhydrostatic, elastic mesoscale model. It uses finite differences and a time-splitting scheme to solve prognostic equations on an Arakawa type-B staggered grid. Its vertical coordinate, though defined as a function of the reference-state pressure, is similar to a terrain-following coordinate. For case studies, MM5 employs observed wind, temperature, and humidity as the initial and boundary conditions and incorporates realistic topography and sophisticated physical processes to represent the appropriate forcing for the development of the observed weather system. These physical processes include clouds, long- and shortwave radiative processes, and surface fluxes of heat, moisture, and momentum. A more detailed description of MM5 is provided by Chen and Dudhia [4], Chen et al. [5], Dudhia [6], and Grell et al. [7] .

The NCAR MM5 modeling system consists of six programs: TERRAIN, REGRID, RAWINS, INTERP, GRAPH, and MM5 [4]. Each of the programs is executed in the above order interdependently. They are composed of a series of scripts that traditionally have been time consuming to modify and execute; recently, however, this process has been simplified by the use of the Java Regional Climate

Workbench, developed at Argonne by Mathematics and Computer Science Division staff, including John Taylor, Veronika Nefedova and Kevin Reitz [8].

To facilitate research collaboration, we have developed a Web-based application tool that enables access via a Web browser to the output of regional climate model runs using the MM5 system. Figure 1 illustrates a typical session. The Web browser uses the native MM5 data format, thus avoiding the need to store duplicate copies of model output, and works efficiently with gigabytes of data. The Web tool was developed by using IDL/ION software. An enhanced version of this Web tool is currently under development.

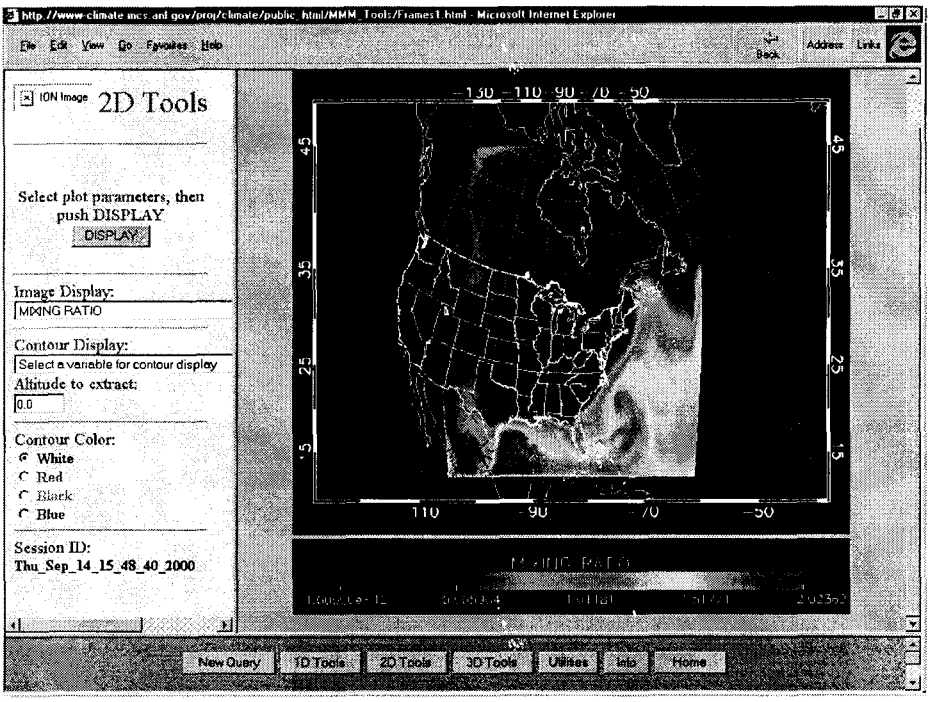


Fig. 1. Web interface displaying the water vapor mixing ratio data from a "Hurricane Huron" simulation. All plots are computed online as required.

3 "Hurricane Huron" Event

An intense cutoff low developed over the Great Lakes region during the period 11-15 September 1996.The system eventually developed an eye over Lake Huron and spiral convection bands producing intense rainfall and wind speeds in excess of 75 mph. While over Lake Huron the low pressure system intensified, with lake surface temperatures observed to fall 4-5° C during this period. Given the similarity in appearance in satellite photographs to a hurricane and to the process of development to a hurricane, this unique Great Lakes weather event was termed "Hurricane Huron" [9].

4 Model Results

We have performed preliminary model runs in climate mode using the latest release of the MM5 modeling system (V3.4) looking at extreme events. Mesoscale resolution climate models provide a consistent framework for us to investigate the link between incoming solar radiation, climate and extreme weather. A series of experiments were undertaken in order to illustrate the importance of enhanced model resolution to simulating the weather, climate, and atmospheric transport processes that will affect extreme weather events.

In this paper we report the results of our model simulations of "Hurricane Huron." We have performed four simulations for the period 6-15 September 1996, at a range of model grid resolutions, 80, 40, 20, and 10km over an identical 2000x2000 km region centered over Lake Huron. By performing our model simulation over an identical spatiotemporal domain we can study the effect of grid resolution on the evolution of the model simulation. We use NCAR/NCEP Reanalysis Project wind fields to provide boundary and initial conditions.

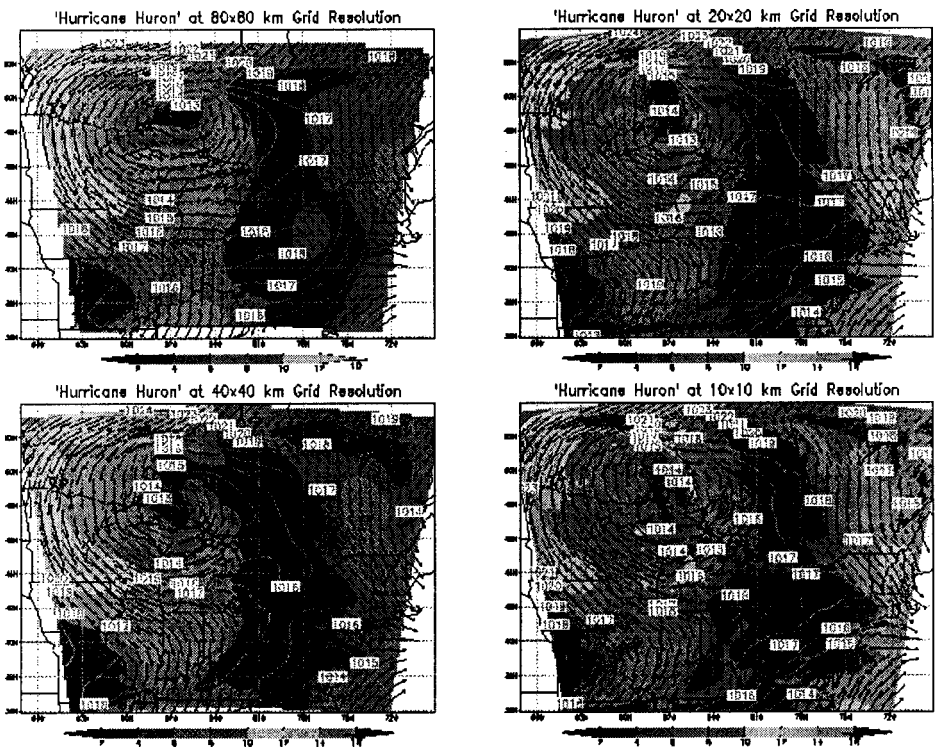


Fig. 2. Wind speed and surface pressure for hour ending 12Z 11 September 1996 at 80, 40, 20, and 10 km grid resolution. Wind speeds intensify, and the formation of a better-defined eye of the low-pressure system is evident as we go to higher model grid resolutions.

Figure 2 illustrates the results of modeling an intense cutoff low that developed over the Great Lakes region during the period 11-15 September 1996. In the model simulations the low-pressure system eventually developed an eye, with spiral convection bands producing intense rainfall and high wind speeds. The color contour intervals show that the wind speed and the contour intervals are identical for all simulations. Arrows represent wind speed and direction and, for clarity, have been plotted at the 80 km grid resolution only, for all plots. Wind speeds increase by up to a factor of 2 as we increase grid resolution from 80 to 10 km. Figure 3 illustrates that hourly rainfall intensity increases dramatically, by nearly an order of magnitude, as we go to higher model resolutions. The pattern of rainfall also changes from broad-scale, low-intensity rainfall at 80 km grid resolution to high-intensity rainfall with significant spatial structure associated with formation of rain bands.

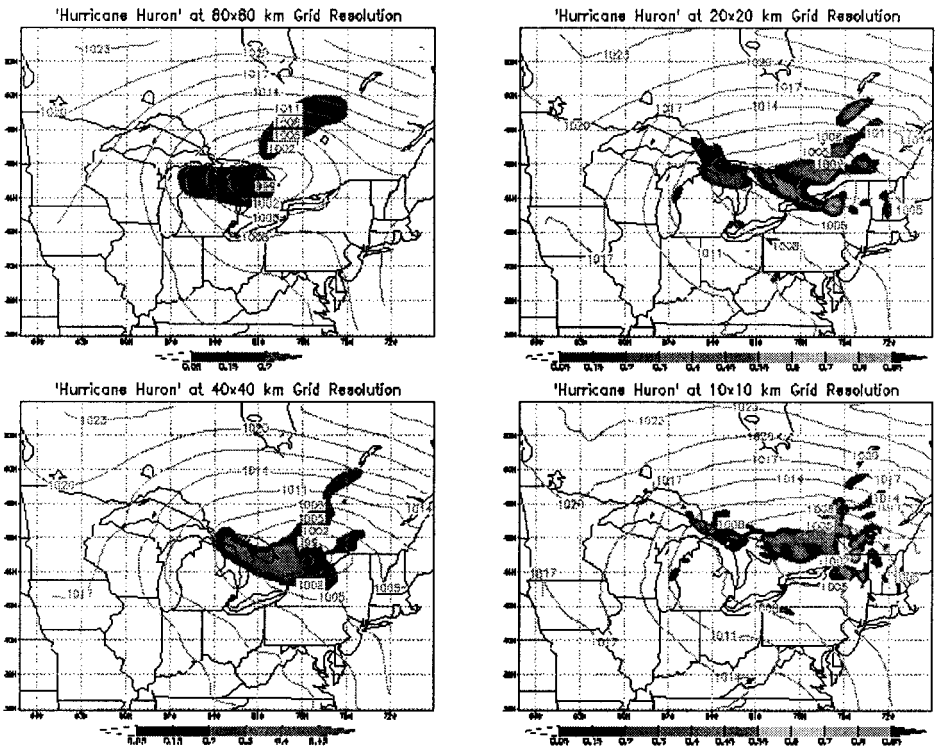


Fig. 3. Precipitation and surface pressure for hour ending 12Z 11 September 1996 at 80, 40, 20, and 10 km grid resolution. Precipitation intensifies by nearly an order of magnitude as we go to higher model resolutions and occupies a smaller more sharply defined rainbands. Increasing precipitation intensities will alter the faction of rainfall allocated to storage in the soil (i.e., soil moisture and runoff), which in turn will alter rates of decomposition and photosynthesis, particularly under water-limited conditions.

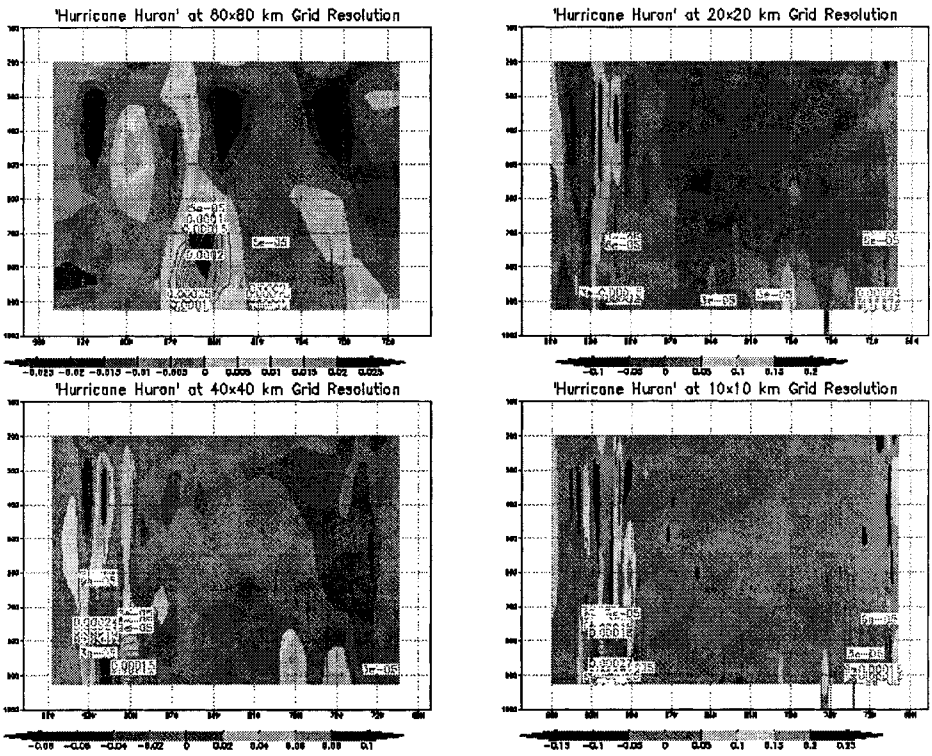


Fig. 4. Vertical velocities ($hPas^{-1}$) for the hour ending 12Z 11 September 1996 at 80, 40, 20, and 10 km grid resolution at 45° N. Vertical velocities intensify by more than an order of magnitude and penetrate to much greater height in the atmosphere as we go to higher model resolutions, and they occupy a more sharply defined regions.

We note that the scales on the graphs in Fig. 4 increase by an order of magnitude in going from 80 to 10 km grid resolution. Figure 4 indicates that changes in vertical velocities associated with the higher model resolution probably plays an important role in the simulation of the precipitation events presented in Fig. 3. As with precipitation, vertical velocities increase dramatically with increasing grid resolution. We also see the appearance of greater structure in the vertical motions in the atmosphere, in that broad-scale vertical motions at 80 km grid resolution are replaced by much more sharply defined, intense vertical motions associated with zones of strong convergence and divergence. This implies that at higher resolutions we are able to better simulate the formation and evolution of rain bands typically associated with such extreme events as "Hurricane Huron." The substantial differences in vertical motions between the simulations at different grid resolutions also has important implications for the study of atmospheric chemistry and to the application of inverse methods to the study of the sources and sinks of greenhouse gases, where vertical motions help determine the concentration of trace gases.

5 Conclusion and Future Research

We have performed preliminary model runs at 80, 40, 20, and 10 km grid resolution in climate mode using the latest release of the MM5 modeling system (V3.4) simulating the extreme Midwest weather event "Hurricane Huron." We conclude that model resolution has played an important role in determining the key parameters typically used to determine the impact of extreme weather events, such as wind speeds and precipitation rates. Model resolution was also found to have a significant impact on atmospheric vertical motions, affecting precipitation rates and their spatial distribution. Changes in vertical motions with increasing grid resolution could also play a significant role in simulations of atmospheric chemistry and inverse modeling aimed at determining the sources and sinks of greenhouse gases.

We will continue to address the key scientific and computational issues in regional climate modeling [10] and their importance to simulating the climate of the US Midwest, including

- defining and delivering high-quality data products via the Web,
- improving the performance of long-term regional climate simulations, and
- assessing the quality of GCM inputs and the role of model spinup and climate drift on regional climate simulations.

210 J. Taylor and J. Larson

We also plan to assess the importance of consistent physics, the sensitivity of climate to the lateral boundary conditions, and the effect of two-way nesting. Finally, the model must be enhanced to include better representation of agriculture (particularly corn and wheat in the U.S. Midwest), natural ecosystems, atmospheric chemistry and biogeochemical cycles of the key greenhouse gases, and the role of the oceans and lakes.

Acknowledgments

We thank the many staff at the Mathematics and Computer Science Division at Argonne National Laboratory in Argonne, Illinois, who assisted us in developing the regional climate simulation capability. This work was supported in part by the Office of Biological and Environmental Research, U.S. Department of Energy, under Contract W-31-109-Eng-38.

References

1. IPCC (1998) The Regional Impacts of Climate Change, Cambridge Univ. Press, Cambridge.

2. IPCC WGI (1990) Climate Change: The IPCC Scientific Assessment. R. A. Houghton et al. (eds.), Cambridge Univ. Press, Cambridge, UK.

3. IPCC WGI (1996) Climate Change 1995: The Science of Climate Change R.A. Houghton et al. (eds.).,Cambridge Univ. Press, Cambridge, UK.

4. Chen, F. and Dudhia, J. (2001) Coupling an Advanced Land-Surface/Hydrology Model with the Penn State/NCAR MM5 Modeling System, Part I: Model implementation and Sensitivity, *Monthly Weather Review*, in press. See also Pennsylvania State University / National Center for Atmospheric Research, MM5 Home Page http://ww.mmm.ucar.edu/mm5/mm5-home.html.

5. Chen, F., K. Mitchell, J. Schaake, Y. Xue, H. L. Pan, V. Koren, Q. Y. Duan, K. Ek, and A. Betts (1996) Modeling of Land-Surface Evaporation by Four Schemes and Comparison with FIFE Observations. *J. Geophys. Res.*, 101, 7251-7268.

6. Dudhia, J. (1993) A Nonhydrostatic Version of the Penn State / NCAR Mesoscale Model: Validation Tests and Simulation of an Atlantic Cyclone and Cold Front. *Mon. Wea. Rev.*, 121, 1493-1513.

7. Grell, G. A., J. Dudhia, and D. R. Stauffer (1994) The Penn State/NCAR Mesoscale Model (MM5). NCAR Technical Note, NCAR/TN-398+STR, 138 pp.

8. Taylor, J. (2000) Argonne National Laboratory, Climate Workbench http://wwwclimate.mcs.anl.gov/proj/climate/public_html/climate-workbench.html.

9. Miner, T., P. Sousounis, G. Mann, and J. Wallman, "Hurricane Huron," *Bulletin of the American Meteorological Society*, February.

10. Giorgi, F., and L. O. Mearns (1999) Introduction to Special Section: Regional Climate Modeling Revisited, *J. Geophysical Research*, **104**, 6335-6352.

Visualizing High-Resolution Climate Data

Sheri A. Voelz[1] and John Taylor[1, 2]

[1]Mathematics & Computer Science, Argonne National Laboratory, Argonne, Illinois 60439
[2]Environmental Research Division, Argonne National Laboratory, Argonne, Illinois 60439
{voelz, jtaylor}@mcs.anl.gov
http://www-climate.mcs.anl.gov/

Abstract. The complexity of the physics of the atmosphere makes it hard to evaluate the temporal evolution of weather patterns. We are also limited by the available computing power, disk, and memory space. As the technology in hardware and software advances, new tools are being developed to simulate weather conditions to make predictions more accurate. We also need to be able to visualize the data we obtain from climate model runs, to better understand the relationship between the variables driving the evolution of weather systems. Two tools that have been developed to visualize climate data are Vis5D and Cave5D. This paper discusses the process of taking data in the MM5 format, converting it to a format recognized by Vis5D and Cave5D, and then visualizing the data. It also discusses some of the changes we have made in these programs, including making Cave5D more interactive and rewriting Cave5D and Vis5D to use larger data files. Finally, we discuss future research concerning the use of these programs.

1 Introduction

Weather simulations are important in helping us understand the key components that determine our weather, particularly extreme events. Manipulating the variables within the simulations gives additional insight into how a particular weather pattern is produced. Unfortunately, many variables are involved in these simulations, making the output files large. For example, one day's data in v5d file format can be around one gigabyte. This number can vary depending on the number of variables the file contains, the frequency with which they are written, and the size of the grid.

Two programs that have been developed for visualizing meteorological data are Vis5D and Cave5D. Vis5D takes atmospheric data values and visualizes them. One can easily view multiple variables at the same time to see how they relate and interact with one another. Vis5D can be run on many different platforms, including Linux, Sun, SGI, and HP. Cave5D is a version of Vis5D for virtual reality environments such as the CAVE and the ImmersaDesk. By visualizing in a virtual environment, a researcher can much more rapidly develop an understanding of the model results. A virtual environment offers much greater freedom to explore and manipulate the data. For example, one can enlarge and rotate the data at much larger scales, in order to study the dynamics of a particular area. Thus, Cave5D makes scientific analysis far easier.

V.N. Alexandrov et al. (Eds.): ICCS 2001, LNCS 2073, pp. 212-220, 2001.
© Springer-Verlag Berlin Heidelberg 2001

2 MM5 Data Format

The MM5 modeling system consists of six programs: TERRAIN, REGRID, RAWINS, INTERP, GRAPH, and MM5. Each of the programs is executed in the above order interdependently. They are composed of a series of scripts that traditionally have been time consuming to modify and execute [1]; recently, however, this process has been simplified by the use of the Climate Workbench, developed at Argonne by Mathematics and Computer Science Division staff, including Veronika Nefedova, John Taylor, and Kevin Reitz [2].

The TERRAIN program defines the domain and map projection, generates the terrain, calculates vegetation and soil categories, and calculates the map-scale factors. The REGRID program calculates first-guess pressure level fields and calculates map-scale factors and the Coriolis parameter. RAWINS is usually used in forecasting and is not used in simulations of past events over long durations. It uses the first-guess fields combined with radiosonde and surface observations to improve the initial state of the simulation. INTERP uses the interpolated pressure-level data calculated in the REGRID program and transforms these data to the model's sigma coordinate. The resulting output from REGRID provides the initial and boundary conditions for an MM5 run. The GRAPH program generates simple line plots from the output of the other programs. Finally, MM5 integrates the mesoscale model for the time period selected producing a series of output files recording the state of the model at fixed time intervals [1].

3 Vis5D

The Vis5D program was developed at the University of Wisconsin-Madison Space Science and Engineering Center, with main contributions from Bill Hibbard, Johan Kellum, and Brian Paul [3]. The Vis5D program provides an interactive environment where users can view five-dimensional gridded data files. The dimensions consist of latitude, longitude, height, time, and an index into a set of physical fields (for example, wind, pressure, temperature, moisture). The most common types of data sets that are viewed using Vis5d are weather and ocean models [3].

3.1 How Vis5D Works

The Vis5D program accepts the v5d and comp5d data formats. MM5 model output data is not in this format. It can, however, be converted from the MM5 format by using Vis5D's import option, NCAR's TOVis5D tool [1], or a similar tool created by the user. The converted data is in a five-dimensional rectangle. The conversion process also creates additional variables needed to run Vis5D. These variables keep track of the number of time steps, variables for the number of rows and columns, information about the grid, the variable names, time stamps, and date stamps [3].

Vis5D starts by reading in the data file. In versions lower than 4.0, the entire file must be loaded into memory. Later versions discarded this restriction and now allow users to run files larger than the allocated size of physical memory. Using virtual memory, Vis5D loads only those data that are needed and discards least-

recently-used data. While the program is displacing data, it performs computations necessary to display isosurfaces, horizontal and vertical slices with contours or in color, volume renderings, and trajectories [3].

3.2 How Vis5D Is Used

Vis5D allows the user to customize the image displayed, while viewing the data. At the command line prompt, many different options can be set. The program allows the user to change window sizes, the way the date is displayed, the amount of memory the program uses, the frame rate, the way the program is loaded, how the display looks, and so on. Most of the arguments can also be changed while the program is running. Selecting the display button of the control panel allows the user to change the default values.

When Vis5D is loaded, the control panel and the display window pop up. Within the control panel the user can change views, chose what data to display and how to display it, animate or step through the data, add a map or topography information, import new data into the current session, save pictures, save configurations, and restore data. An example of the Vis5D interface showing results from our 40 km run of the "Perfect Storm" is shown in Figure 1. The run was computed with MM5v3.4 for October 26, 1991, through November 4, 1991.

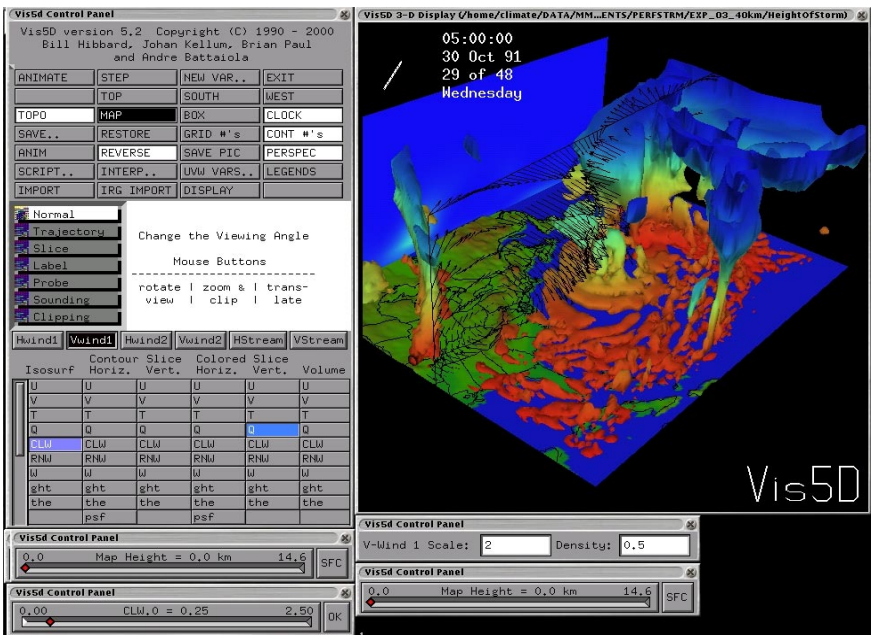


Fig. 1. Vis5D interface displaying data from the "Perfect Storm." The screen shot was taken from our simulation of 5:00 A.M., October 30, 1991.

The Vis5D package also comes with other utility programs to assist in managing Vis5D data files. V5dappend appends v5d data files together. Up to 400 time steps can be appended together without code modification. The v5dinfo tool outputs information about the v5d file. It generates information on the file format, compression size, header size, number of variables, information about the variables and time steps, and the coordinate location of where the data is taken from. Vis5D includes the program v5dstats. This utility computes the minimum, maximum, mean, standard deviation, and number of missing values for each variable in each time step. The package also includes v5dedit, which allows the user to edit a v5d file easily. The user can change a variable's name or unit, the file's times, dates, projection, vertical coordinate system, and low levels. Additionally, the Vis5D package includes v5dimport, which can be used to change the data format of a file to the v5d format.

3.3 Benefits of Using Vis5D

Vis5D allows users to visualize data in a straightforward GUI environment. The program gives users freedom to customize the visualization so every aspect of the simulation can be explored. Graphical object settings can be changed and duplicated to add to its flexibility. For example, the user can change the location of a slice and its colors. Isosurfaces can be displayed at different locations within the data. For example, by changing the value of the variable "rain" (measured in grams), the user can direct Vis5D to draw an isosurface where that particular amount of rainwater is found; higher values will be enclosed within this isosurface. This flexibility is excellent for testing hypotheses relating to a model simulation.

The visualization is presented in a clear, straightforward manner similar to that of a standard weather map. One can add land topography and a polygon map of the region to help locate areas of special interest. Moreover, the isosurfaces, slices, and volumes visualize the data in a simple manner that is easy to comprehend.

3.4 Modifications to Vis5D

Modifications were needed in Vis5D in order to allow us to work with large data sets. In particular, we needed to work with a 1.7-gigabyte v5d file, considered significantly larger than the default settings within the Vis5D code. This file held data from a simulation of the Great Midwest Flood of 1993. The main activity of the flood was during the months of June and July. The model results output by MM5 contained two months of hourly data. In this case, straightforward modifications to Vis5D involved changing the maximum number of time steps VISD5D could handle and increasing the amount of memory it used.

4 Cave5D

Cave5D uses Vis5D code to read in the v5d data file and to compute the desired objects. The basic difference between the two is that Vis5D was written to run on a desktop PC, whereas Cave5D was written for virtual reality environments. With

Cave5D one has the same freedom to change the configuration of the data (although it is more difficult to do so). The virtual environment has the advantage, however, that one can view the data within a larger area than a desktop computer and with higher resolution, thus revealing much greater detail in the model simulation. The results also can be rotated in space, and the size of the image can be changed along the *x*-, *y*-, and *z*-axes. All of these features help scientists to better analyze weather patterns.

The main contributors to the development of Cave5D are Glen Wheless and Cathy Lascara (Center for Coastal Physical Oceanography, Old Dominion University) and Bill Hibbard and Brian Paul (Space Science and Engineering Center, University of Wisconsin) [4].

4.1 How Cave5D Is Used

Even though Cave5D is based on Vis5D, the programs are run differently. Cave5D requires a configuration file, which it reads as ASCII characters so it can be modified using any text editor. This file must contain the path of the v5d input file. Other settings are based on the user's preferences. For example, graphical objects can be added. To generate the graphical objects in a desired form, the Cave5D developers suggest that the user first view the data in Vis5D, then select the settings that look best and save the file as a *.tcl file that can be read by a text editor. This file contains the values that were displayed in Vis5D. These values can then be used in the configuration file [4]. Other options that can be set include memory size, rate, and the display's size, position and color.

Once the Cave5D configuration file is prepared, the program can be loaded and run. (We have written sample instructions for running Cave5D at Argonne.) To bring up the starting panel, the user simply presses the right button on the CAVE wand. The panel allows the user to change the dimensions, animate the data, to speed or slow the animation, and return to the first time step. This panel also contains a button that will switch the panel to one where graphical objects are turned on and off and moved. To rotate the view, the user must hold down the middle button while moving the wand.

4.2 How Cave5D Works

Cave5D begins by performing a CAVE configuration. After configuring the CAVE environment, it processes the configuration file. It then picks out the variables and converts all necessary data from ASCII format. It also identifies which data file to use and which topography and map resolution to use, if applicable.

Cave5D then starts the Vis5D function. First it analyzes the machine(s) that it is running on. Vis5D will count the number of CPU's and will fork off an appropriate number of processes to maximize throughout. Then it starts to analyze and compute the necessary functions to display the data. This process includes initializing the topography, map, and data file and extracting key information from the data file on how the data is to be displayed. Some of this information is used in the next call to draw the clock.

After the clock is drawn in memory, Cave5D creates an "object" view. The object view is a rotation strategy used in manipulating the view of the image. This

function also establishes a navigation matrix that is used to transform the image when maneuvering and scaling in the image in the CAVE.

Next Vis5D functions are called again to compute the graphical objects the user defined in the configuration file. Each object's type is identified and handled accordingly. These computations take up most of the loading time. Objects are placed in a queue and computed in order of their appearance in the configuration file. This strategy was adopted in order to improve the run-time performance of Cave5D.

After this step is completed, the interactive menus are drawn into memory as texture maps. In the Cave5D version 1.4, there are two menus the user can interact with. Plotting pixels in memory draws each of these menus. As the user selects options on these panels, vector coordinates are gathered from the wand position to determine which button was pressed.

Finally, Cave5D does some additional CAVE configurations, and then the program goes into its event loop. Within this loop, it continuously calls CAVE functions to locate the wand and head position in order to correctly redraw the data. It also checks for events such as button pressing. Cave5D will continue in this loop until the user selects the "Close" button on the first menu.

4.3 Benefits of Using Cave5D

Cave5D allows users to easily analyze weather simulations in a virtual environment. Such an environment provides many benefits. In the CAVE one can enlarge the data image and rotate the view to best interpret the simulation. By changing the perspective, one can view the data from a distance or stand in it to see greater detail of the process involved.

Cave5D also allows flexibility. By editing the configuration file, the user can add many options to the simulation run. The benefit is that the users can define the objects that they wish to view when the program is loaded. The weakness is that the users cannot change these values while the program is running. We note that this feature has been changed in the modified version of Cave5D developed at Argonne and scheduled for release as version 2.0.

4.4 Modifications to Cave5D

The first change that was made to Cave5D involved rewriting the code to allow Cave5D to handle large data files with many time steps. Specifically, we converted the 32-bit version into a 64-bit version, modifying variables and changing the compilation flags. This allowed us to run data files in the multigigabyte range with the Cave5D application. We also increased the maximum number of time steps. We were then able to run larger data files containing a large number of time steps successfully. The test data set is discussed in the next section.

One other major modification to Cave5D involved making it more interactive. This was achieved by displaying extra panels that can be used to change the configuration values while in the CAVE. Previously the user had to stop the program, edit the configuration file, and restart Cave5D each time a modification was required. The new approach is far more efficient. Figure 2 displays the extra panels

that were added to Cave5D to allow interactive modification to the configuration values.

Fig. 2. These panel menus were added to Cave5D to increase its functionality. In this example we are going to modify the value of the cloud water isosurface.

The new panels are drawn into memory as follows. The first panel displays the list of the graphical objects that were included in the configuration file. Once the user selects the object to be changed, a new panel is displayed where the user can select which variable to change. Once the variable is changed, the final panel is displayed. This panel shows the object and variable names that were selected, along with the current value. After the user enters in a value and hits the "Enter" button, the entire object is sent back to Vis5D to be recomputed. Each time step of the object is placed inside a queue and is recalculated.

The last major modification to Cave5D involved updating the version of Vis5d. Previously Vis5D version 4.3 was used. Our current Cave5D software uses Vis5D version 5.2 to handle the necessary computations of graphical objects and a majority of the memory management.

5 Midwest Flood of 1993

The Midwest Flood caused fifty fatalities and created damages nearing $15 billion. The flood resulted from heavy rains June through July 1993 on already saturated ground from the past winter and spring. During the month of July nine of the states that were affected by the flood saw twenty days of rain, when on average they would receive about eight to nine days [5].

By visualizing weather patterns such as this flood, we hope to gain a better understanding of why these events occur. This particular data set initially was difficult to visualize because of the size of the file (1.7 gigabytes). With the modifications to Vis5D and Cave5D, however, we were able to visualize the flood in its entirety. Figure 3 represents the data that was ran for July 6, 1993, of the Midwest Flood. The visualization was created through Vis5D and displays rainwater isosurfaces colored to moisture and a horizontal wind slice. The visualization shows moisture flowing from the Gulf of Mexico. The moisture is fed into the upper atmosphere by convection, adding more rainwater into the storm over the Midwest.

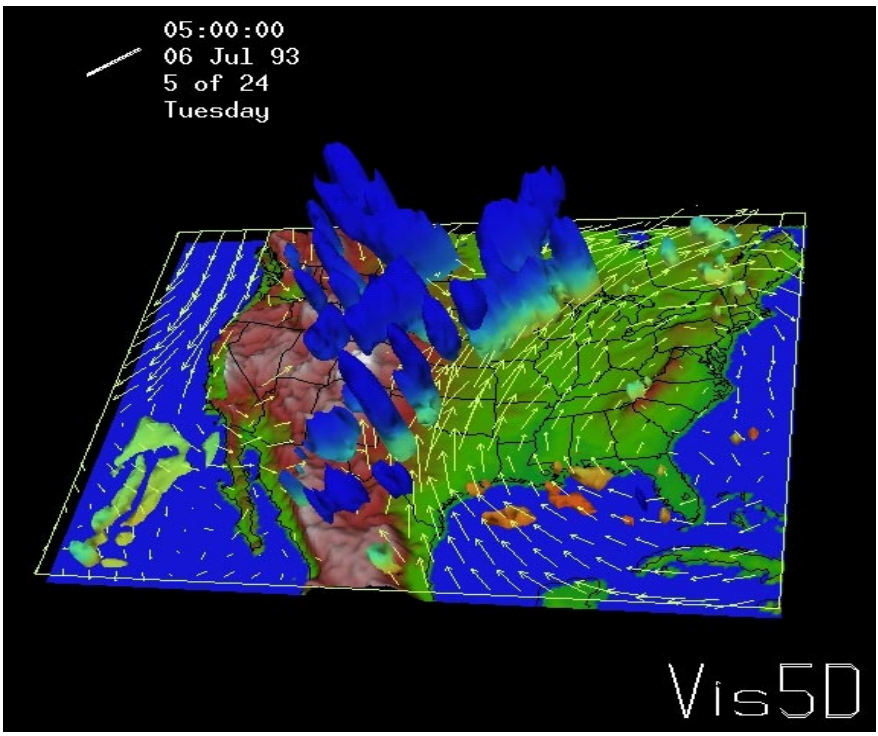


Fig. 3. The Midwest Flood of 1993 on July 6, 1993, at 5:00A.M. These results were part of a 52 km grid resolution MM5v3.4 run.

6 Conclusion and Future Research

Today's technology allows us to run more detailed weather simulation programs, producing even larger data files that are not susceptible to traditional modes of analysis. Visualization, on a desktop or in a virtual reality environment, provides a new way of analyzing these enormous data sets. Our efforts have involved using two new tools, Vis5D and Cave5D, to visualize data from MM5. In addition to converting the MM5 data to a format recognized by these tools, we modified both Cave5D and

Vis5D to handle over a gigabyte of meteorological data. Cave5D was further modified to make it more interactive and user friendly.

Future research will include further modification of bothVis5D and Cave5D. A short-term goal is to be able to work with an entire year of meteorological data. Our long-term goal is to be able to work with terabytes of data. Achieving this latter goal will require rewriting both programs to handle the large amounts of data involved. We also will need to redevelop the way Cave5D computes graphical objects. At present, all graphical objects are computed in their entirety before the program is fully loaded. We need to modify this approach because there is not sufficient memory to hold terabytes of data. Such steps would involve the following:

- Modifying memory management within the Cave5D and Vis5D code to handle the larger terabyte data sets.
- Allowing Cave5D to create graphics within a memory buffer, close to the maximum available memory, to achieve continuous graphics display.
- Modifying Vis5d and Cave5D to enable parallel I/O in order to work efficiently with large data sets and complex visualizations.
- Modifying the sequencing of the graphics calculations in Cave5D.

Acknowledgments

We thank the staff of the Futures Laboratory at the Mathematics and Computer Science Division at Argonne National Laboratory in Argonne, Illinois. This work was supported in part by the Laboratory Director Research and Development funding subprogram of the Office of Advanced Scientific Computing Research, U.S. Department of Energy, under Contract W-31-109-Eng-38.

References

1. Pennsylvania State University / National Center for Atmospheric Research, MM5 Home Page http://www.mmm.ucar.edu/mm5/mm5-home.html, 1999.

2. Taylor, J. Argonne National Laboratory, Climate Workbench http://wwwclimate.mcs.anl.gov/proj/climate/public_html/climate-workbench.html, 2000.

3. Space Science and Engineering Center University of Wisconsin - Madison, Vis5D Version 5.1 ftp://www.ssec.wisc.edu/pub/Vis5D/README, 1999.

4. Old Dominion University, Cave5D Release 1.4 http://www.ccpo.odu.edu/~Cave5D/Cave5DGuide.html, 1998.

5. Larson, L.W. Hydrological Research Laboratory, The Great USA Flood of 1993 http://www.nwrfc.noaa.gov/floods/papers/oh_2/great.htm, 1996.

Improving Java Server Performance with Interruptlets

David Craig, Steven Carroll, Fabian Breg, Dimitrios S. Nikolopoulos, and Constantine Polychronopoulos

Center for Supercomputing Research and Development
University of Illinois
1308 W. Main St, Urbana IL 61801, USA
phone: +1.217.244.4654
{dcraig, scarroll, breg, cdp}@csrd.uiuc.edu

Abstract. With the widespread usage of the Internet, the need for high throughput servers has greatly increased. The Interruptlet system allows Java server application writers to register light-weight interrupt handling routines (written in C or Java). The underlying system architecture is designed to minimize redundant copies between protection domains and thread overhead involved in I/O handling in the JVM on Linux.

1 Introduction

The World Wide Web (WWW), was originally designed as a global document retrieval infrastructure built on top of a network of networks, called the Internet. Documents stored on any node of the WWW can be transferred to any requesting node, using the HyperText Transfer Protocol (HTTP). Documents written in the HyperText Mark-up Language (HTML) can form a system wide interrelated information system that can be retrieved by mouse clicks from a web browser running on a user's desktop.

With the introduction of the Java programming language [3] the web broadened its horizons by allowing more interactive content to be embedded in its documents. Small Java applications, called applets, can now be downloaded to and executed on the same web browser used to access other web documents.

As the Java language grew, so did its attraction as a more general purpose programming language. Many Internet server tasks are now being implemented in Java. Java Servlets provide a mechanism to extend the functionality of web document servers to provide complete e-commerce or virtual community services over the web. Servlets have the potential to replace the more traditional CGI applications, which are less portable and can be harder to program.

Java Servlets are most efficiently employed from within a Java based web server. The Java Virtual Machine (JVM)[4] running the web server can more seamlessly run Servlets than C coded web servers. However, Java based server applications still suffer from low performance[1]. Java based servers, that need to handle high loads, typically employ a large number of threads, introducing a

[1] http://www.volano.com/report.html

V.N. Alexandrov et al. (Eds.): ICCS 2001, LNCS 2073, pp. 223–232, 2001.
© Springer-Verlag Berlin Heidelberg 2001

large overhead. This, combined with high overhead introduced by data transfer from the network device through the operating system to the application, yields less than adequate performance.

This paper presents two contributions. The first is a new form of asynchronous I/O for Linux called an *isocket* (Interruptlet socket) which eliminates much of the context switch and system call overhead involved in the reading and writing of data on I/O ports. The benefits isockets can be exploited by the standard Java I/O library. The second contribution is the concept of a Java *Interruptlet* which leverages the isocket to allow direct interrupt handling code to be registered and invoked for incoming I/O. The handler code interfaces directly with the Virtual Machine and allows the quick handling of requests in a server application.

In Section 2 we describe the design goals for our proposed architecture. Section 3 introduces the design of our architecture. Section 4 shows how our approach can be applied in a typical Java based server application. Section 6 compares our approach to other approaches to improve Java I/O performance. Section 7 presents some future work involving Interruptlet. Section 8 concludes this paper.

2 Design Goals

The current growth in size and popularity of the World Wide Web creates the need for ever more powerful servers to handle more client requests. Faster I/O support is needed to meet this demand, but ideally this faster I/O should be provided in an easily integrated fashion so that existing server software can be improved with little effort. The I/O overhead problem in Java is worse than a C environment because of the added layers of abstraction provided by the virtual machine increase the overhead for each transaction.

In monolithic operating systems, whenever data arrives at the networking device, a DMA transfer is setup to copy that data into a socket buffer in kernel space. Next, the kernel wakes up the process(es) waiting for data on that socket. After a process is woken, it has to transfer the data from kernel memory into its own, requiring expensive protection domain checks and possible page faults. Next, the virtual machine has to wake the thread that is waiting for that data, which requires expensive thread context switching and contending for the virtual machine scheduler lock. A handle to the data which is now in user space is given to that thread which can begin processing.

The motivation for our design was reducing the number of context switches and system calls necessary in a typical transaction for a server application written in Java. Secondly, we wanted to add fast interrupt handler-like routines to Java without exposing the details of the operating system or other non-portable details to the server application written in Java.

3 Interruptlets Design

In this section, we describe our proposed architecture for improved signal handling in Java programs. We will first give a high level overview of the complete process involved, and then explain each component of the architecture in more detail. We will also outline the changes to the JVM that were necessary to support Interruptlet execution. An overview of the architecture is shown in Figure 1.

An Interruptlet is a Java class with a routine to handle a particular type of I/O request. The object is registered with the JVM which then redirects I/O requests to that handler. Currently, short network requests are the primary requests handled by Interruptlets. These are requests such as retrieving a static HTML page or updating counters for the number of bytes received and sent. The primary class of applications that will benefit from Interruptlets is server applications that frequently perform operations that complete in a 100's of cycles.

For example, in the web server application that we will describe in the next section, a commonly occurring set of static web pages is often served. These pages can be cached and served quickly as soon as a request is received by the Interruptlet. Any request that can be handled by the Interruptlet's handling routine is said to have taken the *fast path*. If the Interruptlet cannot handle the request, the request is queued and normal program operation for handling that type of request is resumed. This is referred to as the *slow path*.

To provide fast interrupt delivery from the operating system to the Interruptlet, we add a data structure called an isocket to the Linux kernel. An isocket allows a user process to read incoming data from an area in memory that we call the I/O *shared arena*. The I/O shared arena is a segment of memory shared between the kernel and a user process, like the JVM. The shared arena concept was first used as part of the nanothreads implementation [2]. Because the page is locked in memory and its location is known to both the kernel and JVM, it can be used to exchange information via reads and writes. This avoids the heavy overhead involved in a system call to read data from kernel memory.

The complete Interruptlet architecture consists of a modified Linux kernel, a modified version of the JVM, and a user level Interruptlet library. We will now describe each of these components.

3.1 Linux Kernel Modifications

An isocket is derived from a traditional Linux socket. In a traditional socket, the data that is received is copied from kernel memory to user memory with a system call after the user process is awakened. In an isocket, the data is instead copied network interface and the I/O Shared arena as indicated at location (1) in Figure 1.

If the user process (JVM) is running when new data arrives, the process is sent a SIGIO interrupt (2). Upon receiving this interrupt, it checks the I/O shared arena for the data and receives a pointer to it if there is data available (data may not end up in the arena if it was full). This process is called a fast read

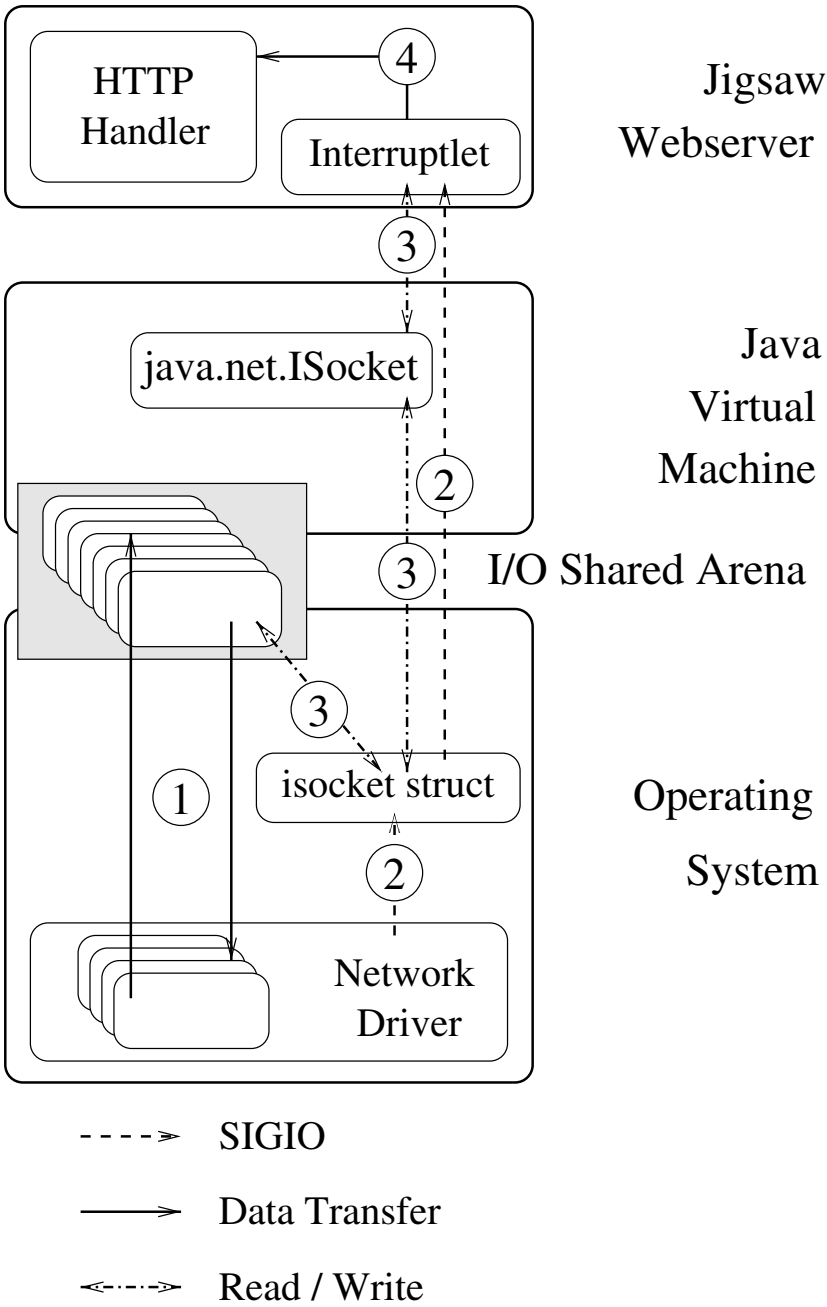


Fig. 1. Interruptlet Architecture

(3) and it saves a system call and the subsequent data copy that is implicit in a normal socket read operation. The data copy time can be hidden with multiple processors, since the copy is no longer coupled with the read call. In addition to reducing the copy overhead, we borrow an already running JVM thread to run the Interruptlet to save the overhead of awakening a thread to handle the I/O as done in the unmodified JVM. The JVM passes the pointer to the data in the I/O shared arena directly to the Interruptlet and invokes its handler routine with that data pointer as its argument. Note, that the Interruptlet code masks all other interrupts while it runs, which poses some restrictions on Interruptlet code, which we will describe in the next section.

Because the JVM can be interrupted by the OS at any time, special care has to be taken when writing the interrupt handler itself. The garbage collector may have been in the process of moving an object when the interrupt occurs, so accessing such objects requires special attention.

The isocket data structure has an identical interface to Linux sockets with two additions: `read_fast()`, and `write_fast()`. The standard socket `read()` and `write()` calls trigger the normal copy from the kernel buffer to the user buffer. In the isocket version, the data is stored in the shared arena on arrival and a ready bit is set. When the `read_fast()` function is called, it polls the ready bits for all of the buffers in the shared arena and selects the first buffer it finds that is ready. A pointer to this buffer is returned. A `write_fast()` operation reserves a buffer in the shared arena and then copies the data to be written there. The kernel then copies it directly to the network adapter from the shared arena (1).

3.2 JVM Modifications

The JVM needs to provide a mechanism to allow Java applications to create isocket structures. A copy of the existing socket classes (`java.net.Socket`, `java.net.ServerSocket`, etc.) was created whose native functions map the Java operations to isockets instead of sockets. In addition, isocket based implementations of the `java.net.SocketInputStream` and `SocketOutputStream` are provided. In addition, the JVM must be set up to invoke application provided Interruptlets whenever it receives a signal from the kernel to do so. This was accomplished by modifying the standard JVM I/O handler to check for registered Interruptlets before running the old handler. Finally, it is necessary to provide a mechanism for registering the Interruptlet itself. The Interruptlet class library provides a static (native) routine for registering an Interruptlet for a particular type of I/O.

3.3 User Level Interruptlets

The final component of the Interruptlets architecture is the language level Interruptlet classes that the user can subclass in order to write handler routines for I/O. The Java Interruptlet object must be of a type that extends the `Interruptlet` class:

```
public abstract class Interruptlet {
    protected Interruptlet(IServerSocket s);
    public abstract byte[] handleInterrupt(byte[] data);
}
```

The Interruptlet constructor takes care of registering the Interruptlet for the communication port associated with the supplied server socket. The JVM invokes the `handleInterrupt()` method whenever the networking device signals the arrival of data. The data parameter contains the incoming data on the networking device. The reply generated by the Interruptlet will be returned in a byte array by the `handleInterrupt()` method. When data is received and the Interruptlet is invoked, the data that is returned by the handler is then immediately sent to the requester.

If the Interruptlet cannot handle the request in a sufficiently short time span, the request is deferred to the slow path by returning a null pointer instead of a byte array handle. The slow path (4) must be made runnable by the Interruptlet by notifying a new thread of control to handle the request. In other words, the slow path is equivalent to how the request would have been handled in the absence of the Interruptlet. Since all interrupts are masked during the completion of an Interruptlet, it is key to quickly return from the Interruptlet handler as soon as possible if the request cannot be handled.

4 Application: Web Server

One of the most important servers found on the Internet is the web server. A web server uses the HyperText Transfer Protocol (HTTP) to serve a wide variety of documents from the server's host to requesting clients. For heavily visited sites, it is important for a web server to maintain a high rate of serviced documents and to provide clients their expected response times.

The typical mode of operation of a web server, like that of many other server applications, is to have a main routine listen and accept connections from client applications and then to spawn a new thread to handle this request. Handling a typical request involves reading and interpreting the request, finding or constructing the requested document, and finally generating a response containing the document and return it to the client.

Jigsaw[2] is a fully customizable web server developed by the World Wide Web Consortium[3]. Jigsaw includes support for CGI and Servlets and is completely written in Java. It largely follows the mode of operation described above.

We plan to add an Interruptlet based cache facility to the Jigsaw web server to allow more efficient handling of static HTTP requests. Based on observations of real world web server statistics, the designers of the SpecWeb99[4] benchmark suite estimate that 70% of all HTTP requests are static HTTP requests.

[2] `http://www.w3.org/Jigsaw/`
[3] `http://www.w3c.org`
[4] `http://www.specbench.org/osg/web99/`

Requests that can be satisfied by this cache constitute an Interruptlet fast path as described in the previous section. The complete architecture of our Interruptlet enhanced Jigsaw web server is shown in Figure 1. The slow path, using the default Jigsaw request handling routines, is taken if the requested document is currently not in the cache.

The Interruptlet cache is a simple Least Recently Used (LRU) cache implemented in native C code that is linked with Jigsaw using the Java Native Interface (JNI). It stores document contents associated with simple HTTP GET request strings. A Java interface to the cache is provided to allow the Interruptlet slow path to update the cache, in addition to the updates from the fast path. We will discuss fully Java written Interruptlet routines when discussing future work in Section 7.

The main loop (accept connection and spawn new handler thread) has to be modified to make our Interruptlet mechanism work. Instead of creating a standard Java `ServerSocket`, we create an `IServerSocket` and register our Interruptlet with it. In addition, we create a pool of handler threads that stand by to execute slow path requests. It is important to note that the modifications necessary to adopt the Jigsaw web server to work with Interruptlets were minimal. The slow path of the server simply reuses the code that was already written for normal handling. The ability to reuse the core of the code is one of the strengths of our approach.

5 Preliminary Results

The Interruptlets system is still in the implementation phase. However, we have conducted experiments to characterize the performance of the isockets part of the design. The test application was a simple echo server that accepts messages of varying lengths and then echoes the same message back to the sender. Two versions of the echo server were created: one version with a standard Linux socket implementation (Linux 2.2.19) and one version with isockets. The isocket version does not currently use the Interruptlets ability to run on the currently executing thread.

In the first experiment, the client and server were run on the same machine, a 4 processor Dell server with Pentium Pro 200Mhz processors. We recorded the time between the client sending the request to the server and the client receiving the reply. The isocket version was about 5% (20–50 μs) faster than the version with normal sockets. Next, the client and server were run on separate machines connected by a cross Ethernet cable. The isocket version was 3% (0–30 μs) faster.

The performance gain is due to the removal of the buffer copies from the critical path. In the normal sockets version, data arrives at the NIC triggering an interrupt. The kernel copies the data from the NIC to kernel space and then notifies the user application that the data is available. The user application then issues a read system call which copies the data to user memory. In the isockets version of the echo server, the interrupt arrives and the kernel copies the data to the shared arena. The kernel notifies the user application that the data is

available and the user application reads it directly from the shared arena. In short, there is one less copy and one less system call.

We expect significant gains in throughput (in terms of number of client requests) from the increased level of concurrency available from the I/O shared arena. The I/O shared arena enables the overlapping of the copying of incoming data to user space with the processing of data in the server. Additional gains are expected from the use of Interruptlets, which reduce the number of context switches required to handle data and reply to the clients.

6 Related Work

The POSIX standard provides its own functions that implement asynchronous I/O. An application calls `aio_read()` to read from a file descriptor, but does not want to be blocked. The application can choose to be signaled when the read operation can be completed or to wait for incoming data with a blocking call at a later time. In standard Linux, this functionality is implemented using separate threads to handle the request. SGI's KAIO[5] uses a more efficient split-phase I/O, where the request is queued at the I/O device.

The Non Blocking I/O (NBIO) library[6], part of the Sandstorm project [8], provides non blocking I/O facilities to Java applications. NBIO is implemented as a JNI wrapper around the native non blocking I/O facilities `select()` and `poll()`.

Interruptlets also provides a mechanism to asynchronously handle incoming data within a Java application. In addition, isockets allow a more efficient propagation of the incoming data through the kernel to the application, by using the I/O shared arena to reduce context switching and protection domain checking overhead.

IO-Lite [6] has a similar approach to our shared arena to minimize unnecessary and redundant buffering and copying. They have a single copy of each I/O buffer that does not need to be copied from kernel space to user space, but they use a read only buffer system instead of a locked page specifically for communication. However, IO-Lite does not reduce context switching overhead.

U-Net provides a zero-copy interface and was the basis for the Virtual Interface Architecture (VIA). It achieves this by providing a user-level interface to the network adapter that allows applications to communicate without operating system intervention. The main distinction between U-Net and Interruptlets is that Interruptlets also tries to eliminate context switching overhead by hijacking the currently running thread to service the interrupt. Also, the combination of the reply with the return from the Interruptlet is unique to our design.

The Flash [5] web server combines a single threaded event driven architecture for cached workloads with a multithreaded architecture for disk-bound requests. Exploiting IO-Lite in their Flash web server improved its performance with

[5] `http://oss.sgi.com/projects/kaio/`
[6] `http://www.cs.berkeley.edu/ mdw/proj/java-nbio/`

40-65%. Redhat TUX web server[7] obtains high performance by moving HTTP handling into the Linux kernel. Our approach is more general in that isockets and Interruptlets are concepts that could be exploited in a wide range of server applications.

7 Future Work

A prototype Interruptlet system is currently being developed and performance characterization is the next logical step. The most important work will be characterizing the length of time the interrupt handler routine can be allowed to execute before the masking of interrupts degrades server performance and reliability. Because performance of the handler routine is critical, the Interruptlet handler code should be statically compiled in advance and dynamically linked at registration time. At present, this is accomplished by writing the cache and handler routine in C using JNI. In the future, these functions will ideally be written in Java and aggressively compiled by a static Java compiler (such as GCJ [1] or the PROMIS compiler system [7]).

8 Conclusion

The Interruptlet system provides Java programs with the ability to register interrupt handler routines (written in C or Java) and link them seamlessly with their Java server applications. The isocket subsystem provides improved I/O for the Java Virtual Machine by eliminating the redundant copying of data between the network adapter, kernel space, and user space. The system eliminates thread wake-up overhead by borrowing the current running JVM thread to run the handler routine. By writing server applications to make use of the Interruptlet user library, the programmer is, in effect, writing completely portable interrupt handlers.

References

1. P. Bothner. A Gcc-based Java Implementation. In *IEEE Compcon*, February 1997.
2. D. Craig and C. Polychronopoulos. Flexible User-Level Scheduling. In *Proceedings of the ISCA 13th International Conference on Parallel and Distributed Computing Systems*, August 2000.
3. J. Gosling, B. Joy, and G. Steele. *The Java Language Specification*. The Java Series. Addison-Wesley Developers Press, 1996.
4. T. Lindholm and F. Yellin. *The Java Virtual Machine Specification*. The Java Series. Addison-Wesley Developers Press, 1996.
5. V.S. Pai, P. Druschel, and W. Zwaenepoel. Flash: An efficient and portable Web server. In *Proc. of the 1999 Annual Usenix Technical Conference*, Monterey, CA, June 1999.

[7] `http://www.redhat.com/products/software/ecommerce/tux/`

6. V.S. Pai, P. Druschel, and W. Zwaenepoel. IO-Lite: A Unified I/O Buffering and Caching System. In *Proc. of 3rd Usenix Symposium on Operating Systems Design and Implementation*, New Orleans, LA, February 1999.
7. H. Saito, N. Stavrakos, S. Carroll, C. Polychronopoulos, and A. Nicolau. The Design of the PROMIS Compiler. In *Proceedings of the International Conference on Compiler Construction*, March 1999. Also available in "Lecture Notes in Computer Science No. 1575" (Springer-Verlag).
8. M. Welsh, S.D. Gribble, E.A. Brewer, and D. Culler. A Design Framework for Highly Concurrent Systems. Technical report, Computer Science Division, University of California, Berkeley, April 2000.

Protocols and Software for Exploiting Myrinet Clusters

P. Geoffray[1], C. Pham, L. Prylli[2], B. Tourancheau[3], and R. Westrelin

Laboratoire RESAM, Université Lyon 1
[1]Myricom Inc., [2] ENS-Lyon, [3] SUN Labs France
Congduc.Pham@ens-lyon.fr

Abstract. A cluster, by opposition to a parallel computer, is a set of separate workstations interconnected by a high-speed network. The performances one can get on a cluster heavily depend on the performances of the lowest communication layers. In this paper we present a software suite for achieving high-performance communications on a Myrinet-based cluster: BIP, BIP-SMP and MPI-BIP. The software suite supports single-processor (Intel PC and Digital Alpha) and multi-processor machines, as well as any combination of the two architectures. Additionally, the `Web-CM` software for cluster management that cover job submissions and node monitoring is presented as the high-level of the software suite.

1 Introduction

In the past 5 years, there has been a tremendous demand, and offer, on cluster architectures involving commodity workstations interconnected by a high-speed network such as Fast Ethernet, Gigabits Ethernet, Giganet, SCI and Myrinet. These architectures are often referred to as Network Of Workstations (NOW) or high-performance clusters (HPC). Several research teams have launched projects dealing with NOWs used as parallel machines. Previous experimentations with IP-based implementations have been quite disappointing because of the high latencies of both the interconnection network and the communication layer. Therefore, the goal of most research groups is to design the software needed to make clusters built with commodity components and high-speed networks really efficient. The NOW project of UC Berkeley [1] was one of the first projects.

The performances one can get on a cluster heavily depend on the performances of the lowest communication layers. Previous experiences have shown that efficient communication layers and fast interconnection networks must be present altogether to build a high-performance cluster. The availability of HPCs, at an affordable price, and adequate communication software is a great opportunity for the parallel processing community to bring these techniques to a larger audience.

In this paper we present a software suite for achieving high-performance communications on a Myrinet-based cluster: BIP, BIP-SMP and MPI-BIP. The software suite supports single-processor (Intel PC and Digital Alpha) and multi-processor machines, as well as any combination of the two architectures. It can

V.N. Alexandrov et al. (Eds.): ICCS 2001, LNCS 2073, pp. 233–242, 2001.
© Springer-Verlag Berlin Heidelberg 2001

be viewed as a collection of highly optimized components that contribute at each level of the cluster communication architecture to provide the maximum of performance to the end-user. It is also possible for the end-user to choose at which level he wants to program, knowing as a rule of thumb that the lowest level usually provides high-performances but less functionalities. Each of these components has been previously described in the literature so the main motivation for this paper is to present to the parallel computer users a bottom-up approach for efficiently exploiting on a daily basis a Myrinet cluster. Of course, the problems covered by cluster-based computing is much broader than the communication-oriented problems we focused on. Issues such as cluster management, check-pointing and load-balancing are also very important. In many cases, these features are desperately lacking on cluster-based environments, as opposed to traditional (expensive and mainly proprietary) parallel computer environments. In this first attempt, however, we will address more specifically the high-performance communication issues as we believe that this point may drive the first motivation to move from massively parallel computer to a cluster-based solution. Additionally, the `Web-CM` software for cluster management that cover job submissions and node monitoring is presented as the high-level of the software suite.

The rest of the paper is organized as follows. Section 2 presents the hardware characteristics of the interconnection network. Section 3 presents the low-level communication layers and Section 4 presents the customized MPI communication middle-ware. Related works are presented in Section 5 and performance measures in Section 6. `Web-CM` is described in Section 7 and we present our conclusions in Section 8.

2 The Myrinet Hardware

The Myrinet communication board uses a PCI slot to connect a node to a Myrinet switch [2]. The bandwidth provided by the network is approximately 160 MBytes/s (1.2Gbits/s), but the PCI bus (32 bits, 33 Mhz) limits the maximum throughput to 132 MBytes/s. All links are full-duplex and the Myrinet switch is a full cross-bar operating a source-based routing algorithm. There are several features that make this kind of technology much more suitable than a traditional commodity network:

- The hardware provides an end-to-end flow control that guarantees a reliable delivery and alleviates the problem of implementing in software the reliability on top of an lossy channel. As message losses are exceptional, it is possible to use algorithms that focus on very low overheads in the normal case.
- The interface card has a general purpose processor that can be programmed in C. The code, called Myrinet Control Program (MCP), is downloaded at the initialization of the board. It is powerful enough to handle most of the communication activity without interrupting the main processor.

– The interface card has up to 8 megabyte memory for buffers (LANai 9). This memory compensates in some cases (contention on I/O bus, on network...) the throughput difference between the communication components.

3 Low-Level Communication Layers

3.1 BIP

At the lowest level of the communication architecture, BIP (Basic Interface for Parallelism) [3,4] provides an efficient access to the hardware and allows zero memory-copy communications. It has been optimized for low latency, high throughput and a rapid throughput increase. As BIP supplies very limited functionalities, it is not meant to be used directly by the parallel application programmer. Instead, higher layers (such as MPI) are expected to provide a development environment with a higher functionality/performance ratio.

The BIP's API is a classical message-passing interface: it provides both blocking and non-blocking communication (`bip_send`, `bip_recv`, `bip_isend`, `bip_irecv`, `bip_wait`, `bip_probe`...). Communications are as reliable as the network, errors are detected and in-order delivery is guaranteed. BIP is composed of a user library, a kernel module and a NIC program. The key points of the implementation are:

A user level access to the network. Avoiding system calls and memory copies implied by the classical design becomes a key issue: the bandwidth of the network (160 MBytes/s in our case and 132 MBytes/s for the I/O bus) is equivalent to the bandwidth of the memory (300 MBytes/s for memory read and 160 MBytes/s for a copy on a computer with a BX chip set).

Long messages follow a rendez-vous semantic: the receive statement must be posted before the send is completed. Messages are split into chunks and the different steps of the communication are pipelined.

Small messages. Since initializations and handshakes between the host and the NIC program are more expensive than a memory copy for small messages so they are written directly in the network board memory on the sending side, and copied in a queue in main memory on the receiving side. The size of this queue is statically fixed and the upper layers must guarantee that no overflow occurs.

Highly optimized, the raw communication performance for BIP is about $5\mu s$ latency one-way. The maximal bandwidth is 126 MBytes/s for a LANai 4 on a PCI 32 bits 33 Mhz (95% of the theoretical hardware's limit residing, in our case, in the PCI bottleneck). Half of the maximum bandwidth is reached with a message size of 4 KBytes. There is a distinction between small and large messages at 1024 bytes.

3.2 BIP-SMP

In the context of SMP nodes, BIP is not able to exploit all of the hardware performance as only one process per node can gain access to the Myrinet board while the other processors must remain idle for communication. BIP-SMP [5] provides the support of several processes per node. The difficulties in doing so are: (*i*) to manage the concurrent access to the hardware and the Myrinet board and, (*ii*) to provide local communications with the same level of performance as BIP over Myrinet. The key points of the implementation are:

Handling the concurrent access to the network. The concurrent access to the send request queue is managed by a lock. In the current implementation on the Linux OS, the lock uses a specific function provided by the kernel (**test_and_set_bit** in kernel 2.2.x). This function guarantees the atomicity of the memory operation. The cost of the lock operation is small compared to IPC system V locks or the `pthread` library locks. With BIP-SMP two processes can overlap the filling of a send request queue. The only operation that needs serialization is to obtain an entry in the send request queue.

Managing Internal communication. For efficiency reasons, BIP-SMP uses both shared memory to implement mailboxes and direct data transfer from user space to user space. The shared memory scheme moves small buffers with two memory copies but small latency, and the direct copy scheme moves large messages with a kernel overhead but a large sustained bandwidth. The shared-memory strategy needs one queue per communicating peer and the amount of shared memory needed increases by the square of the number of local processes. However, as commodity SMP nodes usually contain 2 or 4 processors, the implementation is justified. The direct-memory copy feature is provided by implementing a Linux kernel module to move data from one user space to another user space.

Multi-protocol layer. Another part of this work is to enable BIP-SMP to simultaneously use both remote communications and local communications while hiding this new feature in the BIP's API. We use two independent pools of receive queues per node: one for the internal communications and the other one for remote communications on Myrinet. Then we can allow the receipt of a message from the Myrinet network and the receipt of a message from another process in shared memory at the same time without any synchronization. The use of BIP-SMP is completely transparent as each process receives a different logical number. Everything else is hidden by the BIP-Multi-protocol layer. Variables are available to provide the information about the location of the other logical nodes, the number of processes on the same physical node, etc.

4 MPI-BIP: The Communication Middle-Ware

MPI-BIP [6] is the privileged middle-ware for the end-user. It is a high performance implementation of MPI [7] for Myrinet-based clusters using the BIP

protocol. The current MPI-BIP implementation is based on the work done by L. Prylli on MPI-GM. Most of the code is now shared between MPI-GM and MPI-BIP.

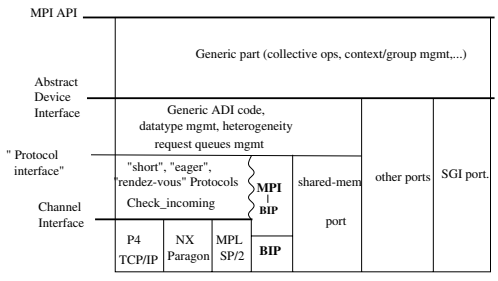


Fig. 1. The architecture of MPI-BIP

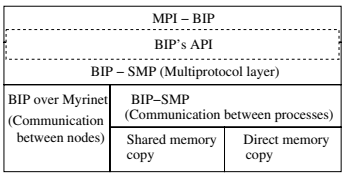


Fig. 2. Architecture of BIP-SMP.

MPICH is organized in layers and designed to facilitate the porting to a new target hardware architecture. Figure 1 presents our view of the MPICH framework and shows at what level we inserted the MPI-BIP specific part. Different ports choose different strategies depending on which communication system they use. We implemented our network specific layer at a non-documented interface level that we will call the "Protocol Interface". This API allows us to specify custom protocols for the different kinds of MPI messages. Each MPI message of the application is implemented with one or several messages of the underlying communication system (BIP in our case). The main contribution of MPI-BIP are:

- As BIP's flow control for long messages relies on the hardware flow-control, it is not sufficient when one side is not able to receive for a long time. For small messages, MPI-BIP uses a credit-based flow control taking into account the size of the BIP's queues.
- MPI-BIP uses request FIFO to allow multiple non-blocking operations.

Figure 2 shows the architecture of the BIP-SMP module within MPI-BIP. The complete view of the communication software architecture can be obtained by replacing the MPI-BIP and BIP blocks in figure 1 by figure 2. We chose to maintain the split view for simplicity.

5 Related Works

First introduced in 1997, BIP was more an incremental step in a large family of software than a complete new design. Especially on Myrinet, we can find many other communication systems: Active Messages from Berkeley[8], Fast Messages from Illinois [9] and U-Net from Cornell University [10]. All these systems bypass the operating system to shorten the communication critical path and to limit, or

avoid completely, memory copies for bandwidth improvements. BIP is however a very efficient implementation.

More recently, the efficient usage of clusters of shared-memory multiprocessors (CLUMPs) has gain attention. We have investigated issues related to a multi-protocol message passing interface using both shared memory and the interconnection network within a CLUMP. Several projects have proposed solutions for this problem in the last few years and BIP-SMP is in the same research line. Projects like MPI-StarT [11] or Starfire SMP Interconnect use uncommon SMP nodes and exotic networks but performances are limited. Multi-Protocol Active Messages [12] is an efficient multi-protocol implementation of Active Messages using Myrinet-based networks and Sun Enterprise 5000 as SMP nodes. Multi-Protocol AM achieves 3.5 μs of latency and 160 MBytes/s of bandwidth. The main restriction is the use of the Sun Gigaplane memory system instead of a common PC memory bus. The polling is also a problem in Multi-Protocol AM as polling for external messages is more expensive than for internal messages. However, Multi-Protocol AM is the first message passing interface to efficiently manage CLUMPs. Finally, one of the first message-passing interfaces to manage CLUMPs as a platform is the well-known device P4 [13] used by MPICH. P4 provides mechanisms to start multiple processes on hosts and uses either message passing or shared memory copies to communicate between these processes. However, the programmer must explicitly select the appropriate library calls. We can also cite implementations of MPI limited to a single SMP node, like the MPICH devices **ch_shmem** or **ch_lfshmem** . The device **ch_lfshmem** is a lock-free shared memory device that achieves very good performance (2.4 μs and 100 MB/s on one of our SMP nodes). Regarding the shared memory management, some works about concurrent access for shared memory Active Messages [14] presents very efficient solutions such as a lock-free algorithm and a high performance lock implementation.

6 Performance Measures

For all measures, the operating system is Linux. Figure 3 and 4 shows the performances of BIP and MPI-BIP for LANai 7 and LANai 9 (the latest board available). The additional cost of MPI-BIP is approximately 4 μs for a 0-byte messages (mainly CPU) over BIP for the latency on our test-bed cluster. Note that the latency of MPI-BIP depends in a large part on the processor speed as the network part of the latency is very small. For instance, as the processor speed is increased, the latency of MPI-BIP is decreased. The results obtained with a LANai 9 on a 2Gbits/s link are still experimental and performed with a back-to-back configuration as no switch was available yet. The jumps in the latency curves come from the BIP distinction between small and large messages at 1024 bytes. For the LANai 9, this distinction is beneficial for the latency. In figure 3, one jump in the MPI-BIP curve comes from BIP; the other one, occurring a bit earlier, comes from the way MPI-BIP switches from short to a three-way eager strategy.

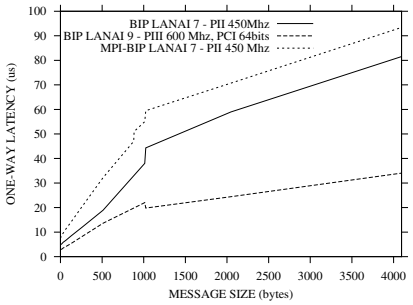


Fig. 3. BIP and MPI-BIP latency.

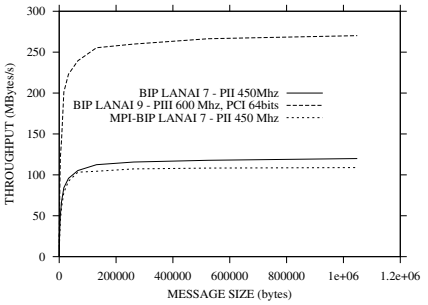


Fig. 4. BIP and MPI-BIP bandwidth.

Table 1 shows the raw point-to-point communications performance of BIP-SMP and MPI-BIP/BIP-SMP. The experimental platform consists in a cluster of 4 dual Pentium II 450 MHz 128 MBytes SMPs interconnected by a LANai 7 Myrinet network. We measured the latency and the bandwidth between two processes using ping-pong communications, on the same node and on two different nodes of the cluster.

Table 1. Pt-to-pt communications with BIP-SMP and MPI-BIP/BIP-SMP

Architecture	*BIP-SMP*	*MPI-BIP/BIP-SMP*
Intra-node latency (Shared memory)	1.8 μs	3.3 μs
Inter-node latency (Myrinet network)	5.7 μs	7.6 μs
Intra-node bandwidth (Shared memory)	160 MBytes/s	150 MBytes/s
Inter-node bandwidth (Myrinet network)	126 MBytes/s	107 MBytes/s

We then compared the latency and the bandwidth of MPI-BIP/BIP-SMP with several other related works: the **ch_shmem** and **ch_lfshmem** MPICH devices; MPICH over GM and MPI-PM/CLUMP (with `mpizerocopy` flag). We used the benchmark program **mpptest** included in the MPICH distribution. This software measures the latency and the bandwidth of a network architecture using a round-trip communication with blocking calls. The tests (cf. figure 5 and 6) are performed by varying the size of the packets sent between two processes on the same node node for the **intra-node** tests and between two processes on two different nodes for the **inter-node** tests.

7 `Web-CM`: Executing Programs on the Myrinet Cluster

A number of steps must be performed before programs can be executed on a Myrinet cluster. These steps are, in a chronological order: install the Myrinet hardware, install the BIP software suite, and run `biproute` to determines the topology of the cluster. We will not describe these steps as they are very specific to the user's hardware and operating system configuration. We will describe

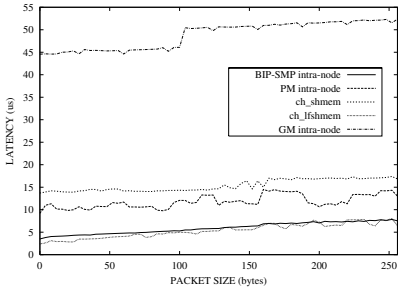


Fig. 5. Intra-node MPI.

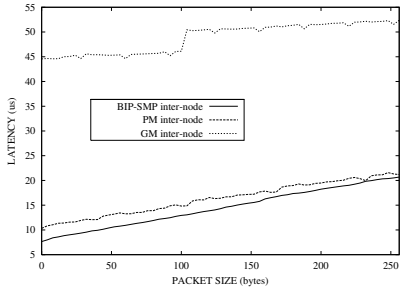


Fig. 6. Inter-node MPI.

below the main steps for running parallel programs on an operational Myrinet cluster: linking the libraries and submitting jobs. Then the `Web-CM` tool is presented.

7.1 Linking the Libraries

Depending on the choice of the end-user, the program may only need the `bip` library, or also the `bipsmp` library if multi-processor support is needed. If MPI is used then the `mpi` library must be added. All the required low-level libraries are automatically included with the `bipcc` command but the user must include the `mpi` library in its `makefile` file. `bipcc` internally calls `gcc` by default but any other compiler can be used (such as g++) with the `-comp` flag.

7.2 Submitting Jobs and Monitoring Nodes

The BIP software comes with a few Perl scripts such as `bipconf`, `myristat` and `bipload` that respectively configures a virtual parallel machine, shows the status of the Myrinet board and launches a program on a virtual machine. For the moment, the utilization of a Myrinet board is exclusive to a user. Therefore, the typical way to submit jobs was to (*i*) run `myristat` to know how many nodes, and which one, are available, (*ii*) run `bipconf` to select the available nodes (if they are in a sufficient number), (*iii*) run `bipconf` to build the virtual machine, and (*iv*) call `bipload` with the program to be executed.

7.3 `Web-CM`: The Integrated Web-Based Cluster Management Tool

`Web-CM` is our first attempt to ease the utilization of a cluster. The main goals of `Web-CM` are to facilitate the submission of jobs and to offer a graphical view of the resources on the cluster. However `Web-CM` must be viewed as an integrated web-based environment and not as a new package for job submission or graphical visualization. `Web-CM` integrates existing packages into the web framework and interacts with them through a number of CGI-bin scripts (mainly Perl and shell scripts).

Fig. 7. Main screen.

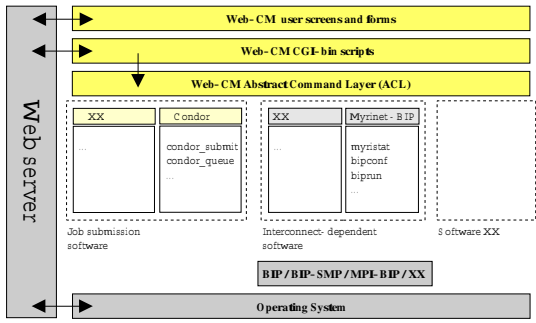


Fig. 8. Architecture of `Web-CM`.

The choice of a web-based environment makes it easy to gain access to remote clusters through a regular web page, and this in the whole Internet. For the moment, it supports Myrinet-based clusters and the `condor` job submission package (`http://www.cs.wisc.edu/condor/`) but an abstraction layer (ACL in fig. 8) makes it possible to adapt it to another type of hardware and software configurations. This ACL simply maps predefined functionalities into specific platform-dependent commands. Such predefined functionalities are for example the list of available nodes, the submission of a job in a batch queue, the status of a job... The realization of the functionality is left to an existing package.

For the moment, `Web-CM` allows a user to graphically view the available nodes and to interactively create a virtual machine by selecting the desired nodes. Then the user can run interactively the job or submit it with the `condor` package. Further implementations will allow an automatic virtual machine configuration just by indicating the number of nodes needed (mainly for queued jobs).

The modular view of `Web-CM`, and the fact that it integrates existing packages rather that developing new ones, allows for an quick increase of its functionalities. For instance, the ACL can be easily configured for another kind of interconnect network technology (SCI, Giganet...) and the new dedicated software package integrated into the common framework. We plan to add additional job submission packages since `condor` was initially chosen because of its free availability but appears to be too complex.

8 Conclusions

HPCs represent a serious alternative to expensive parallel computers. In this paper, we mainly focus on a Myrinet-based cluster and have presented a software suite exploiting on a daily basis such a cluster. The software suite is used intensively in our group for several projects involving genomic simulations and parallel simulations of large scale communication networks. BIP is used worldwide by several universities.

One of the principal aims of this paper is to show the maturity of the more recent technologies. Generalizing the use of the faster systems available will provide to the end-user community a way to reach a higher level of scalability, and to make parallel solutions usable for a wider range of applications, especially those that require fine grain decomposition.

Acknowledgements

The initial version of `Web-CM` was developed by S. Oranger and F. Goffinet (as part of their undergraduate work) and, L. Lefèvre and C. Pham.

References

1. Thomas E. Anderson, David E. Culler, David A. Patterson, and the NOW Team. A case for networks of workstations: Now. *IEEE Micro*, Feb 1995.
2. Nanette J. Boden, Danny Cohen, Robert E. Felderman, Alan E. Kulawik, charles L. Seitz, Jakov N. Seizovic, and Wen-King Su. Myrinet - a gigabit-per-second local-area network. In *IEEE Micro*, volume 15.
3. Loïc Prylli and Bernard Tourancheau. BIP: a new protocol designed for high performance networking on myrinet. In *Workshop PC-NOW, IPPS/SPDP98.*
4. L. Prylli, B. Tourancheau, R. Westrelin, An Improved NIC Program for High-Performance MPI, *in : International Conference on Supercomputing (ICS'99)*, N. P. Carter, S. S. Lumetta (éditeurs), *Workshop on Cluster-based Computing.*
5. P. Geoffray, L. Prylli, B. Tourancheau:, BIP-SMP: High Performance message passing over a cluster of commodity SMPs. *in : Supercomputing'99 (SC99).*
6. L. Prylli, B. Tourancheau, and R. Westrelin. The design for a high performance mpi implementation on the myrinet network. In *EuroPVM/MPI'99*, 1999.
7. W. Gropp, E. Lusk, N. Doss, and A. Skjellum. A high-performance, portable implementation of the MPI message passing interface standard. *Parallel Computing*, 22(6):789–828, September 1996.
8. T. von Eicken. *Active Messages: an Efficient Communication Architecture for Multiprocessors.* PhD thesis, University of California at Berkeley, November 1993.
9. Pakin, S., Karamcheti, V., Chien, A,: Fast messages (FM): Efficient, portable communication for workstation clusters and massively-parallel processors. *IEEE Concurrency*, 1997.
10. von Eicken, T., Basu, A., Welsh, M.: Incorporating memory management into user-level network interfaces. TR CS Dept., Cornell University, 1997.
11. Parry Husbands and James C. Hoe. Mpi-start: Delivering network performance to numerical applications. In *SuperComputing (SC'98)*, Orlando, USA.
12. Steven S. Lumetta, Alan M. Mainwaring, and David E. Culler. Multi-protocol active messages on a cluster of smp's. In *SuperComputing (SC'97).*
13. Ralph M. Butler and Ewing L. Lusk. Monitors, messages, and clusters : the p4 parallel programming system. TR, University of North Florida and ANL, 1993.
14. Steven S. Lumetta and David E. Culler. Managing concurrent access for shared memory active messages. In *International Parallel Processing Symposium*, 1998.

Cluster Configuration Aided by Simulation

Dieter F. Kvasnicka[1], Helmut Hlavacs[2], and Christoph W. Ueberhuber[3]

[1] Institute for Physical and Theoretical Chemistry,
Vienna University of Technology,
dieter@theochem.tuwien.ac.at
[2] Institute for Computer Science and Business Informatics,
University of Vienna
hlavacs@ani.univie.ac.at
[3] Institute for Applied and Numerical Mathematics,
Vienna University of Technology,
christoph@aurora.anum.tuwien.ac.at

Abstract. The acquisition of PC clusters is often limited by financial restrictions. A person planning to buy such a cluster must choose between numerous configurations possible due to the large number of different PC components a cluster may be built of. Even if some of the applications that will be run on the planned cluster are known, it is generally difficult if not impossible to identify the one configuration yielding the optimum price/performance ratio for these applications a priori.
In this paper it is demonstrated how to use the newly developed simulation tool Clue to decide which configuration of the components of a cluster yields the best price/performance ratio for a particular software package from computational chemistry. Due to the simulation based approach, even the impact of components available in the future only can be evaluated.

1 Introduction

Due to the dramatic price decrease of standard PC components and the exponential performance growth as described by Moore's law, traditional supercomputers are gradually being substituted by PC clusters consisting of several standard PCs containing between one and four processors and being interconnected by either Fast Ethernet or gigabit networks.

As there are numerous companies offering off-the-shelf PC components at very different prices and capabilities, a potential cluster buyer is usually faced with a large number of different possible cluster configurations. However, there is usually an upper limit to the overall cluster price, rendering some configurations as being too expensive. An example for a critical and difficult decision is, whether to use the available budget for buying more nodes, more processors per node, more main memory per node or a faster node interconnection network. For selecting one particular configuration it is necessary to know the number of users that are projected to use the cluster and the type of applications that will be run on it.

V.N. Alexandrov et al. (Eds.): ICCS 2001, LNCS 2073, pp. 243–252, 2001.
© Springer-Verlag Berlin Heidelberg 2001

Although there is usually a general understanding of the influence of each component, it is often impossible to judge the total impact the chosen configuration will have on the applications that are foreseen to be run on the cluster. This is even more difficult if the projected applications are parallel applications consisting of several processes run on more than one processor or node. At this point, usually rules of thumb are applied, leaving room for unpleasant surprises once the cluster has been bought and installed.

In this paper it will be demonstrated how to apply the newly developed *CLUster Evaluator* Clue to assess the performance of various cluster configurations for given parallel software, in this case the software package Wien 97 [3], an application code from computational chemistry. This software is available in two different parallel implementations, one requiring a large amount of memory for each processor node, the other one relying on a fast communication network. Both of these requirements nearly double the price of each node.

2 Related Work

In the past, several attempts have been made to simulate the performance of parallel programs.

In trace-driven approaches as carried out for example by the PVM Simulator PS [1], Tau [9] or Dip [8], it is assumed that the interprocess communication patterns, i.e., the number and directions of sent messages, are fixed and do not depend on the run-time situation. This assumption is valid, for example, for routines from the ScaLapack library. In cases where the communication depends on the run-time situation, however, for example when simulating the effect of load balancing mechanisms, in contrast to execution driven simulation, this approach cannot be used.

In case the simulation kernel is execution driven, however, as provided for example by SimOS [10], changing communication patterns may also be taken into account at the expense of increasing the simulation time drastically.

Another approach is taken in the Edpepps [4] tool. Here, users may construct application models for PVM programs by using the graphical program representation language PVMGraph. This representation then drives the simulation kernel. Approaches like this are primarily meant for rapid prototyping, the main drawback being the need for creating program models in addition to the actual implementation.

3 The Simulation Tool CLUE

Being based on the *Machine Independent Simulation System for PVM 3* (Miss-PVM [7], the simulation tool Clue is meant to support (i) configuration decisions concerning clusters of SMPs, (ii) the development of software for parallel computers which are not yet available, (iii) reproducible performance assessments in environments with constantly changing load characteristics (like NOWs), and (iv) the debugging of parallel programs.

Clue allows the simulator user to model cluster configurations by specifying important parameters concerning the communication network and the computational nodes. These parameters are easily obtained either by carrying out measurements on real systems or by taking known or extrapolated parameters (see Section 5.1). The simulator assumes that the simulated parallel applications use the message passing library *Parallel Virtual Machine* (PVM) [6] for communication between the processes. The software models are most easily constructed by taking the original source code as input for the simulator. Thus it is not necessary to rewrite existing C or Fortran code or to create additional code. PVM based code can be used without modification.

The simulator then is driven by actually executing the original parallel program, the simulator routines being activated by catching all calls to PVM and redirecting them to special routines implemented in Clue. The structure of Clue is shown in Fig. 1. At the virtual layer, for each running process one instance of

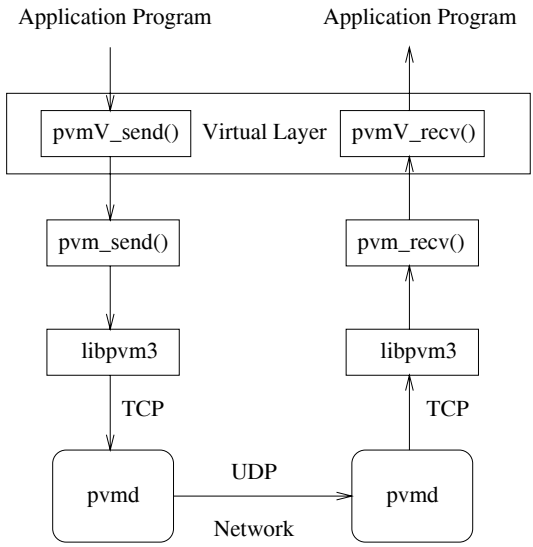


Fig. 1. Structure of Clue.

the simulator is created as well. The simulator itself thus consists of distributed instances communicating with other instances by exchanging PVM messages. Each simulator instance maintains its own local virtual time, representing the simulated computing time of this process. The virtual time, however, must not be confused with the real simulation runtime. Instead, it is a variable controlled by the simulator instances and increased if the respective attached application process has carried out some computation. This is detected by measuring the CPU time consumed by the attached processes between two adjacent calls to PVM. The local virtual time is then increased by this measured CPU time multiplied by the processing time factor of the node hosting the process as specified in the configuration file.

Virtual time is also increased when sending or receiving application messages, as carried out by the executed application program. It must be assured, however, that the reception of messages is in correct order with respect to the global virtual time, i. e., if at virtual time t_1 process P_1 sends a message to P_2 waiting for messages to arrive, it must be impossible that after being woken up and receiving this message, another process P_3 being at virtual time $t_2 < t_1$ sends a message to P_1. Thus, the simulator instances must use a protocol for distributed simulation in order to synchronize their local virtual times and guarantee message delivery in the right order. Therefore, the used protocol relies on an

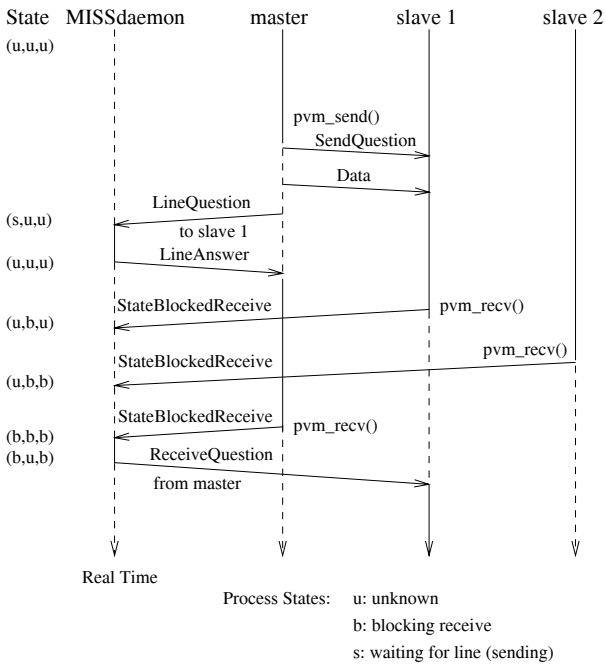


Fig. 2. Distributed simulation protocol.

additionally spawned process called "MISSdaemon" for synchronization. The daemon keeps track of all processes, marking each process to be either in state *unknown*, *blocking receive*, *waiting for line*, *waiting for probe* or *killed*. Fig. 2 shows sample protocol messages in case a master process at the application level sends a message to a slave process. Upon calling a PVM routine, each simulator instance sends its new state to the daemon. As soon as it can be assured that the global virtual time can not be violated, the receiving slave is allowed to receive the message.

Due to the execution driven simulation approach, processor performance and memory access, including cache performance, are modeled quite accurately. Also network properties are modeled with high accuracy, only depending on a few input parameters. Disk I/O, however, is not modeled at all.

4 Hardware Configuration Parameters

The cluster configuration model used by Clue is called a "virtual machine". Virtual machines are defined by an input file being read in at simulation start. This file contains the specification of a machine used for the master program and for additional machines or host types. For each record possible parameters are:

Name of the machine or host type. This name is used in `pvm_spawn`. If the machine *performance factor* (described next) is 0, the machine is assumed to be *real*, and the program is started on this machine. Otherwise the machine has a *virtual* name, and PVM 3 is asked to look for a suitable machine.

Performance factor. This is a floating-point multiplier p for calculating the computation time. If this parameter is 0, the computation timing results are not changed. Otherwise, if a process of a parallel application consumes n CPU seconds between two adjacent calls to the virtual layer, the virtual time is increased by $p \times n$ seconds.

Initialization Time. This is the time needed for `pvm_spawn` to start a new child process, measured on the child's side.

Spawn Time. This is the time spent in `pvm_spawn` by the calling process.

Send Time. This is the time used for sending a message using `pvm_send` or `pvm_mcast`. This time contains packing the message, resolving the address of the host and starting the transmission (as far as the sending process is involved). The parameters may be specified as k, d, where the send time $s(m)$ depending on the message length m is given by $s = k \times m + d$, or may be specified as tuples $(m_x, s(m_x))$, where the actual send time is interpolated linearly between these points.

Receive Time. This is the time used in calling the receive routines `pvm_recv`, `pvm_nrecv` and `pvm_probe`. This time is always the same whether these routines succeed or fail.

Transmission Time. This is the time used to transfer a message minor the send delay. As with the send time, the transmission time may be specified as linear model or linearly interpolated data points.

Packing Time. This is the time used to pack the message into the PVM 3 send buffer.

Various values for time granularity have already been used for simulation. Fast interconnection networks can be modeled accurately with $1\mu s$ granularity. Send and transmission times may be specified for any pair of hosts, they may also be specified for the send and transmission of messages from one host to itself, in case multiprocessor machines are to be modelled. If the actual performance model turns out not to be exact enough, it can easily be changed by modifying the configuration file.

5 Case Study: Finding an Optimum Cluster Configuration for WIEN 97

In this case study it is demonstrated how to apply Clue to find an optimum hardware configuration for a particular parallel application, in this case by taking the well known computational chemistry package Wien 97 as an example. It is assumed that an institute is planning to purchase a PC cluster to run the computationally expensive Wien 97 simulations. Furthermore it is assumed that there is a tight limit on the budget that the institute can spend.

The most time consuming part of Wien 97 is spent in a routine called `lapw1`, basically solving a generalized symmetric eigenproblem by applying Cholesky factorization and transforming the problem to a (simpler) tridiagonal eigenproblem. Thus, a cluster running Wien 97 should be optimized for running parallel versions of the Cholesky factorization and for tridiagonalization, preferably as provided by the basic linear algebra subprograms Blas [5], which are used by the parallel linear algebra package ScaLapack [2].

Also, two application cases have been chosen for simulation:

- A small case with matrix sizes of 2500×2500 which can be solved on one processor on a standard PC.
- A larger case with matrix sizes of 6000×6000. In this case it is necessary to either have more memory for each processor, or to distribute each matrix to several processors.

Each case is representative for 50 % of the overall workload of the PC cluster.

5.1 Communication Models

Though Clue can apply various communication models, in this case study, piecewise linear models are used for the send and transmission times (see Fig. 3). All model parameters have been measured on existing networks by running custom programs for measuring parameters like latency, bandwidth and processing speed.

Additionally, a simple contention model is used which increases the communication time (both send and transmission time) by a factor depending on the number of simultaneous communication operations carried out.

Alternative approaches for estimating configuration parameters include for instance taking published values from benchmarks or parameters provided by companies for their products (often specified, for example, for gigabit networks). Also, a popular rule of thumb specifies that when increasing the processor clock rate by $N\%$, then the performance gain will not exceed $N/2\%$. This rule may then be applied for extrapolating the performance of systems, even if the respective processors are not available yet. However, it does not apply if new processor cores with new features like SIMD instructions are introduced or if other possible performance bottlenecks like caches or main memory are changed.

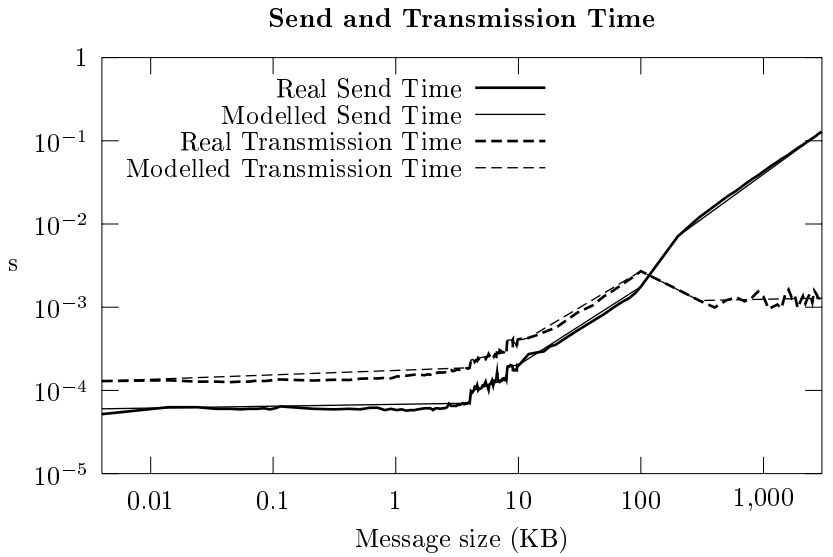


Fig. 3. Send and transmission time for a Fast Ethernet PC cluster. Sender and receiver are on the same node. Although they share the same memory, they communicate via a message passing library.

5.2 Hardware Configurations

To choose the configuration of PC clusters with highest performance for Wien 97, several configurations are examined. The budget limit being set to $20.000,– reflects only pure hardware investments, costs for the cluster installation and configuration as well as for software are not included and are assumed to be similar for all configurations.

Table 1 shows the configurations *affordable* according to the budget limit. In this table, information on the clock rate and the manufacturer were omitted due to the fact that updates on clock rates occur too often and customers will always aim at buying the fastest available version of a processor at the time of purchase. Thus, this decision is delayed to the final stage of the evaluation.

Table 1. Cluster configurations under consideration.

Name	Number of Nodes	Memory per Node	Processors per Node	Interconnection Network
Network	6	256 MB	2	Gigabit Class
Memory	6	1024 MB	2	Fast Ethernet
Fine	7	128 MB	1	Gigabit Class
Coarse	7	1024 MB	1	Fast Ethernet
Cheap	10	256 MB	2	Fast Ethernet
Cheapest	15	128 MB	1	Fast Ethernet

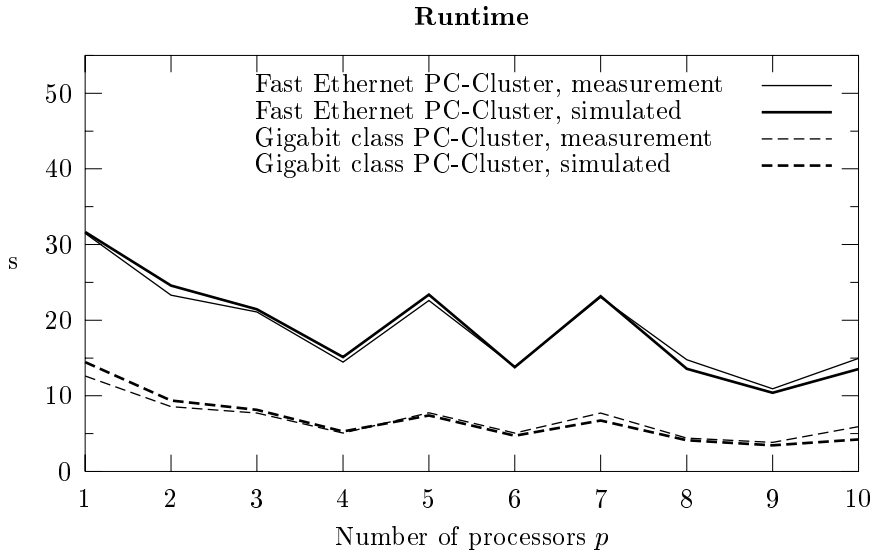


Fig. 4. Simulation of the Cholesky factorization using Clue ($n = 2000$).

6 Simulation Results

Preliminary experiments simulating the Level 3 Blas based Cholesky factorization have been performed with (i) a model for Fast Ethernet communication and (ii) a model assuming gigabit class communication. The communication model was constructed and validated using the PC cluster of the RWTH Aachen using an SCI network. Results with increasing number of processors are given in Fig. 4. It can be seen that for both network types, best results are achieved with a rectangular (e. g., 2×2, 2×3, 2×4 or 3×3) processor grid. In general, empirical observation shows that linear algebra algorithms work best if the processors can be mapped onto a $N \times M, N > 1, M > 1$ processor grid. Thus, most cluster configurations under consideration (with two exceptions) hold this property.

Furthermore, for the small application case it turned out that only the number of processors, their performance, and in the dual processor nodes memory contention have a significant effect on the overall performance, the influence of the network or the memory size being negligible. Thus, instead of applying simulation, an analytical model was chosen to evaluate the configuration performance. The performance $P(C)$ of configuration C is given by $P(C) = N \times f$, where N denotes the number of processors and f denotes the memory contention factor for *dual* processor nodes. This factor is further defined to be $f = 0.5 \times f_c + 0.5 \times f_t$, where $f_c = 1.0$ is the memory contention factor for the Cholesky factorization and f_t being the memory contention factor for the tridiagonalization, which has been measured on a Pentium II system to be 0.77 for *dual* processor and 1.0 for *single* processor nodes.

Results of the final simulation experiments for the large application case are shown in Table 2. In some cases more than one parallelization strategy per configuration is shown. The column *high level* parallelization tells how many instances of `lapw1` are solved concurrently. The column *low level* parallelization tells how many processors are used for each instance of `lapw1` (including shared-memory parallelism). The column *parallelism* tells how many processors are engaged in total (the product of *high level* and *low level*). The column *empty* tells how many processors are not used in the program run. The columns *Kernel 1* and *2* tell the overall floating-point performance in Mflop/s for the Cholesky factorization (*Kernel 1*) and the tridiagonalization (*Kernel 2*).

Table 2. Floating-point performance (in Mflop/s) for the large test case.

Name	High Level	Low Level	Parallelism	Empty	Kernel 1	Kernel 2
Network	3	4	12	0	5194	1578
Memory	3	4	12	0	4153	1501
Memory	3	2	6	6	3066	884
Fine	1	6	6	1	2225	814
Fine	1	7	7	0	1852	702
Coarse	3	2	6	1	2283	1025
Coarse	3	1	3	4	1552	587
Cheap	3	6	18	2	4666	1913
Cheapest	3	4	12	3	3284	1763
Cheapest	3	5	15	0	3248	1543

7 Choosing the Optimum Configuration

In order to find the optimum cluster configuration for Wien 97, the simulation results shown in Table 2 and the results given by the analytical model must further be investigated. This is done by distributing points to each result in such a way that the fasted configuration is rewarded 100 points. The points given to the remaining configurations then show how many percent of the fastest performance they achieved. Furthermore, the points for the large test case are split equally to the two kernels, each being assigned at most 50 points. The result is shown in Table 3, for configurations having two entries in Table 2, only the better entry has been considered.

It turns out that the *Cheap* configuration has to be considered for the final decision, whereas the *Cheapest* configuration being second best. The *Network* configuration performs also well for the large test case and might even win the competition (for the large test case) if communication becomes more important. This may happen either

- if the processor speed increases,
- if less "high level" parallelism is available, or
- if smaller problems have to be solved. In this case the ratio of communication to computation is increased.

The other configurations perform rather poor, mainly because of a lack of raw compute power, since they have fewer processors.

Table 3. Assessment by points.

Name	$P(C)$	Large Case Kernel 1	Large Case Kernel 2	Sum
Cheap	100	45	50	195
Cheapest	85	32	46	163
Network	60	50	41	151
Memory	60	40	39	139
Coarse	40	22	27	89
Fine	40	22	21	83

8 Conclusion

In this paper it has been demonstrated how to use the cluster evaluator Clue for finding optimum PC cluster configurations for given parallel applications and budget limits. By applying this technique, a priori performance evaluations of PC clusters to be bought may be carried out, thus being able to assess different cluster configurations by a quantitative procedure rather than a rule of thumb.

References

1. R. Aversa, A. Mazzeo, N. Mazzocca, U. Villano, Heterogeneous system performance prediction and analysis using PS, IEEE Concurrency 6–3 (1998), pp. 20–29.
2. L. S. Blackford et al., ScaLapack Users' Guide, SIAM Press, Philadelphia, 1997.
3. P. Blaha, K. Schwarz, P. Sorantin, S. B. Trickey, Full-Potential, Linearized Augmented Plane Wave Programs for Crystalline Systems, Comp. Phys. Commun. 59 (1990), pp. 399–415.
4. T. Delaitre et al., A Graphical Toolset for Simulation Modelling of Parallel Systems, Parallel Computing 22–13 (1997).
5. J. J. Dongarra, J. Du Croz, I. S. Duff, S. Hammarling, A Set of Level 3 Blas, ACM Trans. Math. Software 16 (1990), pp. 1–17,18–28.
6. A. Geist, A. Beguelin, J. J. Dongarra, W. Jiang, R. Manchek, V. Sunderam, PVM: Parallel Virtual Machine—A Users' Guide and Tutorial for Networked Parallel Computing, MIT Press, Cambridge London, 1994.
7. D. F. Kvasnicka, C. W. Ueberhuber, Developing Architecture Adaptive Algorithms using Simulation with MISS-PVM for Performance Prediction, Proceedings of the International Conference on Supercomputing, ACM, 1997, pp. 333–339.
8. J. Labarta et al., Dip: A parallel program development environment, Proc. Euro-Par '96, Vol. II, Springer-Verlag, Berlin, 1996, pp. 665–674.
9. W. Mohr, A. Malony, K. Shanmugam, Speedy: An integrated performance extrapolation tool for pC++, Proc. Joint Conf. Performance Tools 95 and MMB 95, Springer-Verlag, Berlin, 1995.
10. M. Rosenblum et al., Using the SimOS Machine Simulator to Study Complex Computer Systems, ACM TOMACS Special Issue on Computer Simulation (1997).

Application Monitoring in the Grid with GRM and PROVE*

Zoltán Balaton, Péter Kacsuk, and Norbert Podhorszki

MTA SZTAKI
H-1111 Kende u. 13-17. Budapest, Hungary
{balaton, kacsuk, pnorbert}@sztaki.hu

Abstract. GRM and PROVE were originally designed and implemented as part of the *P-GRADE* graphical parallel program development environment running on clusters. In the framework of the biggest European Grid project, the DataGrid project we investigated the possibility of transforming GRM and PROVE to a Grid monitoring infrastructure. This paper presents the results of this work showing how to separate GRM and PROVE from the P-GRADE system and to turn them into standalone Grid monitoring tools.

Keywords: Grid monitoring, message passing programs, performance visualisation.

1. Introduction

GRM and PROVE are available as parts of the *P-GRADE* graphical parallel program development environment. *GRM* is a semi-on-line monitor that collects information about an application running in a distributed heterogeneous system and delivers the collected information to the *PROVE* visualisation tool. The information can be either event trace data or statistical information of the application behaviour.

Semi-on-line monitoring means, that any time during execution all available trace data can be required by the user and the monitor is able to gather them in a reasonable amount of time. Semi-on-line monitoring keeps the advantages of on-line monitoring over off-line monitoring. The performance or status of an application can be analysed or visualised during the execution. The scalability of on-line monitoring is better than that of off-line monitoring. Data can be analysed in portions and unnecessary data can be discarded before processing new portions of data. Moreover semi-on-line monitoring puts less overhead on monitoring: trace data are buffered locally and it is sent in larger data blocks than in on-line monitoring. It also stresses the collection site and the trace processing application less than on-line collection since trace is sent only when requested. This way the overload of the collector can be avoided.

PROVE supports the presentation of detailed event traces as well as statistical information of applications. It can work both off-line and semi-on-line and it can be

* This work was supported by a grant of the Hungarian Scientific Research Fund (OTKA) no. T032226

V.N. Alexandrov et al. (Eds.): ICCS 2001, LNCS 2073, pp. 253-262, 2001.
© Springer-Verlag Berlin Heidelberg 2001

used for observation of long-running distributed applications. Users can watch the progress of their application and realise performance problems in it.

P-GRADE is a graphical programming environment integrating several tools to support the whole life cycle of building parallel applications. It provides an easy-to-use, integrated set of programming tools for development of general message passing applications to be run in heterogeneous computing environments. Its main benefits are the visual interface to define all parallel activities in the application, the syntax independent graphical definition of message passing instructions, full support of compilation and execution in a heterogeneous environment and the integrated use of the debugger and the performance visualisation tool. Components of the P-GRADE program development environment are the GRAPNEL graphical parallel programming language, the GRED graphical editor to write parallel applications in GRAPNEL, the GRP2C precompiler to produce the C code with PVM or MPI function calls from the graphical program, the DIWIDE distributed debugger, the *PROVE* execution and performance visualisation tool and the *GRM* distributed monitor. For detailed overview of the tools in P-GRADE, see [6] and [7]. GRM is described in [3], while PROVE is presented in [4]. Further information, tutorial and papers about P-GRADE can be found at [5].

In this paper, we present the problems and implementation issues of the GRM monitor redesigned for semi-on-line general application monitoring in a grid environment. In the next section, we shortly present the original design goals and the structure of GRM. In section 3, we discuss the problems with GRM in a grid environment and present our solutions to these problems. Finally, Section 4 compares GRM with NetLogger.

2. The GRM Semi-online Monitoring Tool

The monitoring in GRM is *event-driven*, both *trace collection* and *counting* are supported. The measurement method is *software tracing* and the instrumentation method is *direct source code instrumentation*. For a classification of monitoring techniques, see [8]. P-GRADE controls the whole cycle of application building, and source code instrumentation is supported by graphics. The precompiler inserts instrumentation function calls into the source code and the application process generates the trace events.

The main goals in the original design of GRM have been strongly related to the P-GRADE environment. The monitor and the visualisation tool are parts of an *integrated* development environment and they support monitoring and visualisation of P-GRADE applications at source level. The monitor is portable among different UNIX operating systems (Irix, Solaris, Linux, Tru64 UNIX, etc.) which is achieved by using only standard UNIX programming solutions in the implementation. GRM is a semi-on-line monitor, that is, the user can let GRM to collect the actual trace data or statistical information about the application any time during the execution. Semi-on-line monitoring is very useful for the evaluation of long-running programs and for supporting debugging with execution visualisation. Both trace collection and statistics are supported by the same monitor and the same instrumentation of the application.

Trace collection is needed to pass data to PROVE for execution visualisation. Statistics mode has less intrusion to the execution by generating fixed amount of data and it supports initial evaluation of long-running applications.

For trace storage *shared-memory segments* have been used on each host for two main reasons. First, semi-on-line monitoring requires direct access to all trace data any time during the execution. The shared buffer can be read by a Local Monitor independently from the application process, when the user asks to collect trace data. Second, if a process aborts its trace data can be saved and analysed to the point of failure.

GRM is a distributed monitor. It consists of the following three main components (see its structure in **Fig. 1**):

Client library (See "Appl. Process" in the figure.)
The application is instrumented with functions of the client. Both trace events and statistics can be generated by the same instrumentation. The trace event types support the monitoring and visualisation of GRAPNEL programs. An instrumented application process does not communicate outside of the host it is running on. It places trace event records or increments counters in a shared memory buffer provided by the Local Monitor.

Local Monitor
A Local Monitor (LM) is running on each host where application processes are executed. It is responsible for handling trace events from processes on the same host. It creates a shared memory buffer where processes place event records directly. Thus even if the process terminates abnormally, all trace events are available for the user up to the point of failure. In statistics collection mode, the shared memory buffer is used to store the counters and LM is responsible for generating the final statistics data in an appropriate form.

Main monitor
A Main Monitor (MM) is co-ordinating the work of the Local Monitors. It collects trace data from them when the user asks or a trace buffer on a local host becomes full. Trace is written into a text file in Tape/PVM format (see [9]), which is a record based format for trace events in ASCII representation. MM also performs clock synchronisation among the hosts.

Both trace collection and statistics are supported by the same monitor and the same instrumentation of the application. Trace collection is needed to give data to PROVE for execution visualisation. PROVE communicates with MM and asks for trace collection periodically. It can work remotely from the Main Monitor process. With the ability of reading new volumes of data and removing any portion of data from its memory, PROVE can observe applications for arbitrary long time.

The integration of GRM into a development environment made it possible to put several functionalities of a stand-alone monitoring tool into other components of P-GRADE. For example:

- instrumentation is done in the GRED graphical editor,

- trace events of different processes are not sorted into time order since the pre-processing phase in PROVE does not need a globally sorted trace file,
- the monitor is started and stopped by GRED,
- Local Monitors are started on the hosts defined by the environment,
- the monitor does no bookkeeping of processes

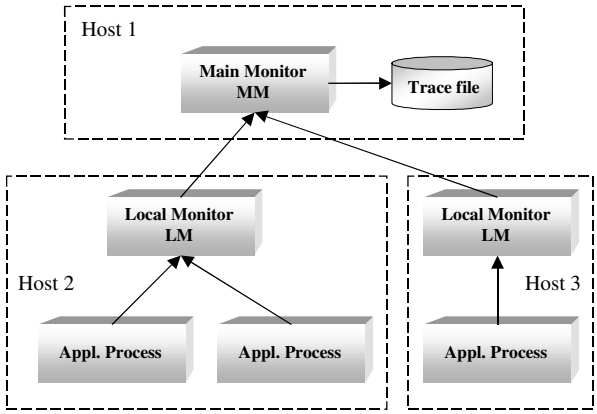


Fig. 1. Structure of GRM

Although the start and exit of the application processes are also events that are put into the trace buffer, monitor processes do no bookkeeping so GRM cannot recognise when the application is finished. The simplest solution was to leave everything to GRED that already knows when the application terminates. GRED sends a message to the MM after program termination and GRM collects the yet uncollected trace from the Local Monitors.

3. Monitoring Applications in the Grid

Monitoring of applications in a grid environment brings new requirements for a monitoring tool:

1. Scalability to a large number of resources and events is very important.
2. The problem of starting up the monitoring system in the grid must be solved.
3. Measurements must have accurate cross-site timestamps.

In addition, the original design goals of GRM must be reviewed. Specifically, two goals must be changed:

1. General application monitoring should be supported (not only GRAPNEL). This also requires user defined event data types.
2. GRM and PROVE should be standalone monitoring and visualisation tools (not part of an integrated development environment).

Other goals are unchanged but they are now requirements not just design goals. Portability remains very important, since the grid consists of heterogeneous resources. Semi-on-line monitoring must be supported, because the grid is a changing environment, where off-line monitoring does not help. Both statistics and event trace collection is needed to monitor large and long running applications that are typical in the grid. Getting trace data to the point of failure is very important, since errors are more frequent in a grid environment. Remote execution of applications from the monitoring data processing site is the normal operation in the grid.

3.1. Scalability

To be usable in a grid environment, a general application monitor must be able to handle a large number of resources and events. Since Local Monitors store event trace in local buffers GRM can already handle large sets of trace data. When the local trace buffer is full, its content should be transferred to some other place to make empty space for further data. To easily recognise the influence of the monitor on the application execution and possibly eliminate it in the statistics, GRM uses a special trace collection scheme in the original design. When a buffer is full all processes of the application are stopped, the Main Monitor collects all trace data from each host and sends the full trace to PROVE or writes it into a global trace file.
The trace collection scheme in GRM is the following:

1. When the buffer on a host is full (more exactly, it is filled up to a predefined threshold is), the process actually recognising this situation notifies the LM.
2. The LM notifies the MM that its buffer needs to be emptied.
3. The MM starts a global trace collection:
   - First it asks Local Monitors to stop the application processes.
   - When all processes are stopped the MM collects the trace data from each LM.
   - MM sends data further to PROVE or writes it into a trace file.
   - MM finishes the collection by notifying Local Monitors, which let the processes continue their execution.

Since the MM collects data from the hosts one after the other and does not merge the traces, we can only ensure that trace events from individual hosts are in time order. The tool processing the trace should sort data itself in a pre-processing phase. PROVE does this pre-processing and this greatly simplifies the Main Monitor.

Stopping all processes before trace collection and restarting them after finishing it ensures that no events are generated by the application during trace collection. As a result, the collection phase can be recognised in the trace (and visualisation), since timestamps are either less than collection start time or greater than collection finish time. This solution makes it possible to eliminate the collection overhead from the statistics.

Unfortunately the above trace collection scheme does not work well in a grid environment. Here stopping all processes might take a long time and it is not scalable for a large number of resources, either. Because of this, the collection scheme must be changed. In a grid, the MM does not initiate a global trace collection when a LM indicates that its buffer is full. Instead, it only collects trace data from this LM.

The current handling of the local buffer (by setting the threshold appropriately) makes GRM scalable to a large number of events. However, its scalability could be further enhanced by modifying Local Monitors to support double buffering or to use local trace files to temporarily store trace data until the Main Monitor collects it.

3.2. Standalone Tool and Start-Up

In the original design, start-up and termination of GRM is controlled by GRED. The Main Monitor of GRM has no information about where to start the Local Monitors. GRED provides all the necessary information since it knows the hosts from a configuration file, and directories containing binaries and their access method (host name, user account, remote shell program name) are defined in dialogs of GRED or in its start-up script. The start-up of GRM in P-GRADE goes in the following way:

- GRED launches the MM with a script (on a remote host or locally), GRED passes its listening socket port number as an argument.
- MM connects to GRED on the given port
- GRED sends information about the execution hosts and the MM launches a LM on each host
- GRED sends the trace file name to the MM
- GRED asks the MM to start the initial clock synchronisation
- MM reports the finish of the clock synchronisation
- GRED starts the application

After starting, the application processes call an instrumentation function that connects them to their Local Monitors. The LM creates a FIFO through which processes can connect to it and get the shared memory segment (and semaphore) identifiers. The name of the FIFO is predefined and it depends only on the user id. Because of this, only one application can be monitored per user. After receiving the shared identifiers, the processes can start generating trace events. The client library functions in the processes recognise if the shared buffer is almost full and tell this fact to the LM. This simplifies the structure of the LM process so it can wait in a *select* system call most of the time and does not consume resources.

Problems with the above if GRM is used as a standalone tool in a grid are:

1. There is no GRED, the user interfaces with GRM directly. Either via the command line or via tools implemented using the monitor control API of GRM.
2. FIFO of the LM should have a name that does not depend on the user id only, since on a grid resource more than one user can be mapped to the same local user id.
3. The Local Monitors cannot be started on a host explicitly, since it is the competence of the local job-manager on a grid resource to decide where jobs will run. This local policy cannot be influenced. Because of this, the Main Monitor cannot start the Local Monitors. It should be prepared to accept connections from them instead.
4. The executable of the Local Monitor should be transferred to the grid resource where the application is run. The easiest way to do this is to link the LM executable to the application as a library. This way we have a single executable

which contains both the application and the LM, and can be started the same way as the application. This solution is independent of any specific grid implementation but requires that the application can be relinked. However, the application has to be instrumented for monitoring with GRM, so this can be assumed.

The start-up of GRM in a grid environment goes in the following way:

- The user launches the MM.
- The MM gives back its port number. The hostname and port number pair identifies this MM.
- The user sets the trace file name (e.g. using the command line interface to the MM).
- The user starts the application that also contains the LM linked in, giving it the MM identifier as a parameter.
- The application is started on some resources in the grid.

After starting, the application processes call an instrumentation function that tries to connect them to the Local Monitor through a FIFO that contains the MM identifier in its name. If a process detects that there is no LM listening on this FIFO yet, it forks, and becomes the Local Monitor. Its child continues as an application process. The LM creates the shared buffer and the FIFO through which processes can now connect to it. To resolve the race condition the processes should wait for a random time interval before trying to connect to the LM. This way the process that picked the smallest wait time will do the fork and create the Local Monitor. In addition, when an LM fails to bind its listening socket to the FIFO, it should free all allocated resources and become inactive, since in this case another process should have already successfully created the LM. When an LM is created, the application processes connect to it and the LM connects to the MM. After successfully connecting to the Main Monitor, the LM notifies the processes. From this point, the processes can start generating trace events.

3.3. Clock Synchronisation

The generated trace can be globally consistent if clock synchronisation is performed regularly and local timestamp values are adjusted to a consistent global timestamp. The offsets of the clocks can be determined by the well-known "ping-pong" message exchange. The message exchange in GRM is done through the sockets connecting Local Monitors with the Main Monitor. After stopping the application processes but before starting trace collection the MM performs the synchronisation with LMs.

This clock synchronisation technique works well on clusters of workstations connected by a LAN but grids require a more sophisticated algorithm. In a grid environment the resources (e.g. clusters) are usually connected by WAN links that have higher latency than the LAN used inside the resource. GRM determines the clock offsets of each LM (running on a host at the remote grid resource) relative to the host of the MM but the accuracy of this measurement is limited by the latency of the WAN link. Because of this, the error of the clock-offset measurement can be comparable to or bigger than the time intervals between events generated at the remote resource (e.g. the start and end of a communication on the LAN).

Since there are several tools (e.g. NTP) that can be used to synchronise clocks, this problem can be solved independently from monitoring. For this reason, GRM does not support clock synchronisation in grid environments, instead it assumes that the clocks are already synchronised.

3.4. User Defined Trace Event Types

The event types originally supported by GRM were only those that are required for monitoring and visualisation of GRAPNEL programs. For general application monitoring, GRM should be re-designed to support arbitrary event types. This is achieved by a two step procedure. First it should be possible to define an event type and then event records of this type can be generated by a simple function call giving event data as function parameters. The client library function

```
int GMI_DefineEvent(int eid, char *format, char *dsc);
```

is provided for defining trace event formats in the application process. A description for the event type can be given in the *dsc* parameter. The *format* parameter should be given in the standard printf format. This is used by the trace generation library when printing an event string. The integer *eid* is the identifier of the event type that can be used in the following function:

```
int GMI_Event( int eid, ...);
```

This will generate a trace event with the parameters given as variable arguments corresponding to the format of this event type. These two functions give a general API for trace event generation. However, for easier use GRM provides additional functions for some common even types. Block *begin* and *end* pairs are very common type of events to identify e.g. procedure calls and exits in the application. Message passing event types are predefined to generate *send* and *receive* events. GRM in the newest versions of P-GRADE already uses this general API to generate GRAPNEL events. In fact, original GRAPNEL event types are internally implemented with the new API and instrumented P-GRADE applications use the original event functions.

4. Comparison with NetLogger

NetLogger is a distributed application, host and network logger developed at the Lawrence Berkeley National Laboratory in the USA, see [1] and [2]. Its main suggested application areas are performance and bottleneck analysis, selecting hardware components to upgrade (to alleviate bottlenecks), real-time and post-mortem analysis of applications and correlating application performance with system information. Its logical structure can be seen in **Fig. 2**, where *netlogd* is a data-logging daemon that writes trace data into the trace file. *Sensors* and *instrumented application processes* can generate events in ULM (Universal Logger Message) format using the client library of NetLogger and send them to the netlogd daemon. Sensors can run remotely from netlogd and send data through the network.

The basic difference between GRM and NetLogger is that GRM uses a shared memory buffer to store trace events locally before sending them to the Main Monitor on another host. In NetLogger, processes should store trace events in their local memory or they should send them right after their generation to the remote collector process.

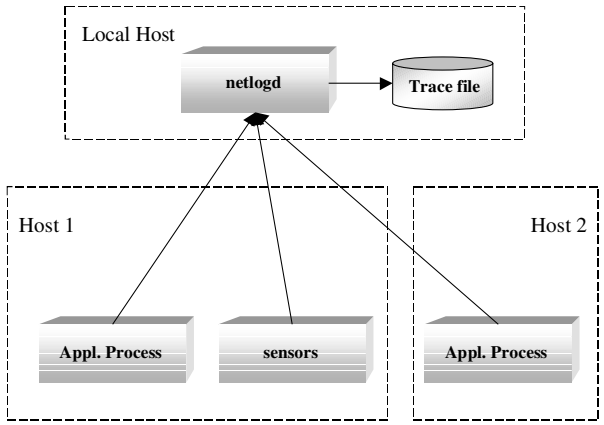


Fig. 2. Structure of NetLogger

While NetLogger's collection mechanism is based completely on the *push* data model, GRM uses a mixed *push/pull* mechanism for the data transfer. In the push model, data is sent to its target without checking the target's availability of receiving. In the pull model, data is stored locally and is sent to a target only for a specific query from the target. GRM works with both mechanisms. When the local buffer is full the Local Monitor sends (pushes) data to the main collector. However, when the buffer is not full yet, the target (e.g. a visualisation tool) can make a request for all available data. In this case, GRM collects (pulls) all available data from the Local Monitors. The advantages of GRM over NetLogger are:

- *Local buffering* of trace data is done independently from the monitored application, so the Main Monitor can collect it any time. In NetLogger data should be sent immediately to the remote *netlogd* process or can be buffered locally but in this case, the visualisation tool must wait until the block of trace data arrives. In GRM the visualisation tool can send a query for trace data any time and the monitor collects data from the Local Monitor processes.
- *Efficient data transfer.* With local buffering in a shared memory segment, application processes and sensors can give trace events to the monitoring tool quickly so they can have very low intrusiveness. Thus, the application runs almost as efficiently as without instrumentation. In NetLogger, the application process is blocked while it sends trace to the collecting process over wide-area network.
- *Scalability.* The use of Local Monitors, local buffering and semi-on-line trace collection mechanism makes GRM more scalable than NetLogger in the sense of number of processes, number of events and event rate.

– *Trace data to the point of failure.* Local buffering in a shared memory segment helps to keep all trace events when a process aborts. Thus, the visualisation tool can show all events until the point of failure.

5. Conclusions

A grid environment brings new requirements for monitoring. The GRM monitor and PROVE performance visualisation tool of the P-GRADE graphical parallel programming environment are good candidates to be standalone grid-application monitoring and performance visualisation tools. We examined their features and monitoring mechanisms and compared them to the requirements of a grid. With some modifications and redesign GRM can collect trace files from large distributed applications in a grid that can be examined in PROVE. In the design of GRM, scalability and problematic start-up issues in grid were considered. The new version of GRM for grid applications will be implemented based on this design.

6. References

[1] D. Gunter, B. Tierney, B. Crowley, M. Holding, J. Lee: "NetLogger: A Toolkit for Distributed System Performance Analysis", Proceedings of the IEEE Mascots 2000 Conference (Mascots 2000), August 2000, LBNL-46269

[2] B. Tierney et al.: "The NetLogger Methodology for High Performance Distributed Systems Performance Analyser", Proc. of the IEEE HPDC-7 (July 28-31, 1998, Chicago, IL) LBNL-42611

[3] N. Podhorszki, P. Kacsuk: "Design and Implementation of a Distributed Monitor for Semi-on-line Monitoring of VisualMP Applications", Proceedings of DAPSYS'2000 Distributed and Parallel Systems, From Instruction Parallelism to Cluster Computing, Kluwer Acad. Publ., pp. 23-32, 2000.

[4] P. Kacsuk: Performance Visualization in the GRADE Parallel Programming Environment, HPCN Asia, Beijing, China, 2000.

[5] P-GRADE Graphical Parallel Program Development Environment: *http://www.lpds.sztaki.hu/projects/p-grade*

[6] P. Kacsuk, G. Dózsa, T. Fadgyas, and R. Lovas: "The GRED graphical editor for the GRADE parallel programming environment", FGCS journal, Special Issue on High-Performance Computing and Networking, Vol. 15 (1999), No. 3, April 1999, pp. 443-452.

[7] P. Kacsuk: "Systematic macrostep debugging of message passing parallel programs", FGCS journal, Special Issue on Distributed and Parallel Systems, Vol. 16 (2000), No. 6, April 2000, pp. 609-624.

[8] J. Chassin de Kergommeaux, E. Maillet and J-M. Vincent: "Monitoring Parallel Programs for Performance Tuning in Cluster Environments", In "Parallel Program Development for Cluster Computing: Methodology, Tools and Integrated Environments" book, P.Kacsuk and J.C.Cunha eds, Chapter 6., will be published by Nova Science in 2000.

[9] É. Maillet: "Tape/PVM: An Efficient Performance Monitor for PVM Applications. User's guide", LMC-IMAG, Grenoble, France, 1995. Available at http://www-apache.imag.fr/software/tape/manual-tape.ps.gz

Extension of Macrostep Debugging Methodology Towards Metacomputing Applications

Robert Lovas[1], Vaidy Sunderam[2]

[1]MTA SZTAKI Computer and Automation Research Institute,
Hungarian Academy of Sciences
P.O. Box 63, H-1518 Budapest, Hungary
rlovas@sztaki.hu

[2]Emory University, Dept. of Math & Computer Science
1784 N. Decatur Rd. #100
Atlanta, GA, 30322, USA
vss@mathcs.emory.edu

Abstract. This paper focuses on the non-deterministic behaviour and architecture dependencies of metacomputing applications from point of view of debugging. As a possible solution we applied and also extended the macrostep systematic debugging methodology for metacomputing applications. Our extended methodology is based on modified collective breakpoints and macrosteps furthermore, we introduce host-translation tables generated automatically for exhaustive testing. The prototype is developed under the Harness metacomputing framework for message box communication based applications. The main implementation issues as well as the architecture of our systematic debugger are also described as the further development of X-IDVS Harness-based metadebugger.

1 Introduction

Debugging of metacomputing applications can be much more exhausting task contrary to debugging of sequential or even parallel programs. This problem comes from the following features of metacomputing: (i) heterogeneity, (ii) dynamic behaviour of computational environment, (iii) large amount of computational resources, (iv) authorisation/authentication on different administration domains, (v) non-deterministic execution of metacomputing applications. During our previous debugging project [15] we have already given some efficient solutions for (i)-(iv) but the *systematic handling of non-determinism* was out of scope of that work. In this paper[1] we focused on the issues of the non-deterministic behaviour of metacomputing applications caused by the varying relative execution speeds of tasks as well as the architecture dependent failures. For instance, it seems a given metacomputing

[1] The work presented in this paper was supported in part by U.S. Department of Energy grant # DE-FG02-99ER25379 and National Research Grant (OTKA) registered under No. T-032226.

V.N. Alexandrov et al. (Eds.): ICCS 2001, LNCS 2073, pp. 263–272, 2001.
© Springer-Verlag Berlin Heidelberg 2001

application always generates correct results on a particular architecture or a combination of architectures (where the programmers originally developed their application) *but* often fails on other architectures. Mostly, the reason for this behaviour is the varying relative speeds of tasks together with the hazardous and untested race conditions. Besides, these different timing conditions might be occurred more frequently in metacomputing environment than in case of dedicated clusters or traditional supercomputers because of the different implementation of the underlying operating systems/communications layers and the unpredictable network traffic, CPU loads or other dynamical changes. By metacomputing applications the above described phenomenon can be very crucial because we cannot ensure that our metacomputing application always runs on the same nodes with almost the same timing conditions.

The only way to prove the 'metacomputing-enabled' feature of an application is the usage of *systematic* testing methods in order to find the timing/architecture dependent failures in the implemented code. For this purpose we applied and also extended the macrostep systematic debugging methodology that has been introduced originally for message passing parallel programs developed by P-GRADE graphical programming environment [10]. Our prototype is under development in the Harness metacomputing framework [14] and we also applied the achievements of the earlier developed X-IDVS metadebugger tool [15].

This paper is organized as follows. In the next section we introduce briefly the Harness framework, the Java Platform Debugger Architecture (JPDA) and X-IDVS metadebugger tool as the basis of our prototype. Section 3 describes the fundamental principles of the extended macrostep debugging methodology and some implementation details. Finally, Section 4 summarizes our project and points out the most current related work.

2 Background

2.1 Harness Metacomputing Framework

Harness attempts to overcome the limited flexibility of traditional software systems by defining a simple but powerful architectural model based on the concept of a software backplane. The Harness model consists primarily of a kernel (see Figure 2) that is configured, according to user or application requirements, by attaching "plug-in" modules that provide various services. Some plug-ins are provided as part of the Harness system, while others might be developed by individual users for special situations, while yet other plug-ins might be obtained from third-party repositories.

By configuring a Harness distributed virtual machine using a suite of plug-ins appropriate to the particular hardware platform being used, the application being executed, and resource/time constraints, users are able to obtain functionality and performance that is well suited to their specific circumstances. Furthermore, since the Harness architecture is modular, plug-ins may be developed incrementally for emerging technologies such as faster networks or switches, new data compression

algorithms or visualization methods, or resource allocation schemes - and these may be incorporated into the Harness system without requiring a major re-engineering effort.

The fundamental abstraction in the Harness metacomputing framework is the *Distributed Virtual Machine* (DVM) (see Figure 1, Level 1). Any DVM is associated with a symbolic name that is unique in the Harness name space but has no physical entities connected to it. *Heterogeneous Computational Resources* may enroll into a DVM (see Figure 1, Level 2) at any time however, at this level the DVM is not ready yet to accept requests from users. To get ready to interact with users and applications the heterogeneous computational resources enrolled in a DVM need to load *'plug-ins'* (see Figure 1, Level 3). A plug-in is a software component implementing a specific *service*. By loading plug-ins a DVM can build a consistent service baseline (see Figure 1, Level 4). Users may reconfigure the DVM at any time (see Figure 1, Level 4) both in terms of computational resources enrolled by having them join or leave the DVM and in terms of services available by loading and unloading plug-ins.

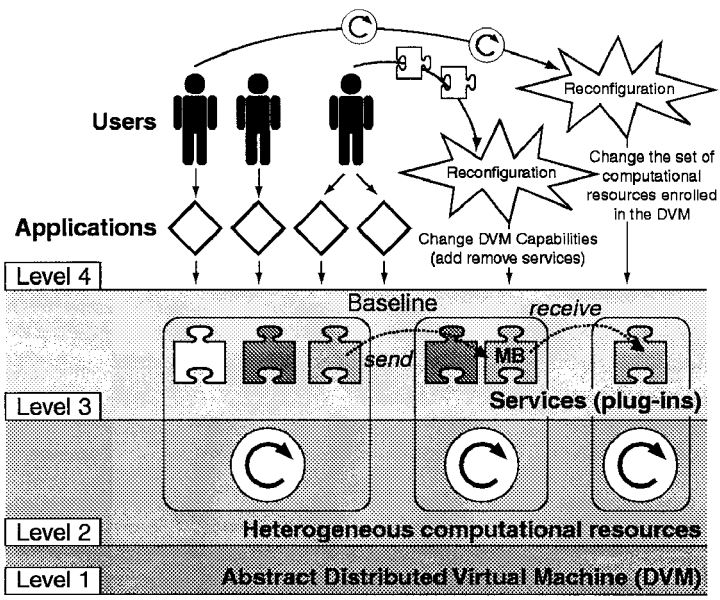


Fig. 1. Abstract Model of a Harness DVM with Message Box (MB) Service

The availability of services to heterogeneous computational resources derives from two different properties of the framework: the portability of plug-ins and the presence of multiple searchable plug-in repositories. Harness implements these properties mainly leveraging two different features of Java technology. These features are the capability to layer a homogeneous architecture such as the Java Virtual Machine (JVM) over a large set of heterogeneous computational resources, and the capability to customize the mechanism adopted to load and link new objects and libraries.

2.2 Java Platform Debug Architecture

Java Platform Debug Architecture (JPDA) is available for almost all widespread platforms as part of Java SDK 1.3. In outline, JPDA provides a high-level remote debugging interface for debuggers called Java Debug Interface (JDI). For the purpose of out-of-process debugging, JPDA gives the Java Virtual Machine Debug Interface (JVMDI) to the debuggee/target JVM. Between the JDI and the JVMDI, the Java Debug Wire Protocol (JDWP) is responsible for transporting both debug requests and debug events. Hence, JPDA can form a base of our X-IDVS debugger (see Section 2.3) by its remote debugging facilities (see Figure 2 between JVM1 and HMCPI).

2.3 Extendible Integrated Debugger & Visualization Service for Harness

In order to solve the emerging debugging issues in the field of metacomputing we already defined the fundamental principles of an extendible, programmable and integrated debugging & visualization tool [15]. The next target was to design and implement a prototype; X-IDVS (eXtendible Integrated Debugger & Visualization Service) applying the defined principles and relying on the Harness framework as well as the above described Java Platform Debugger Architecture.

In order to illustrate briefly the novelty of this work, the main features of the current X-IDVS prototype can be summarized as follows; X-IDVS was designed as a *real metacomputing application* itself hence, the debugger tool can adapt totally to the debugged application and also can take all advantages of the metacomputing environment, such as fault-tolerance, dynamic behaviour, support for heterogeneous computational environment and authorization. When a plug-in is loaded by the user's application anywhere in the metacomputer, X-IDVS can load and activate some system plug-ins on the target host for debugging/monitoring purposes (using the same authorization keys as the loaded plug-in). Moreover, for providing efficient debugging support for RMI-based plug-ins, X-IDVS offers some unique debugging capabilities for RMI communication. Firstly, during step-by-step execution X-IDVS is able to hide the differences between the traditional and remote method invocations from user's point of view. Basically, it means two automatic context switches during an RMI call (client to server/server to client side). On the other hand, X-IDVS combines some program visualization techniques with debugging methods. Hence, the user can get a big picture about the history of plug-ins with the help of an integrated semi-online visualization tool depicted the communication interacts among Harness plug-ins.

Another significant feature of the system is the *extendibility*. X-IDVS can invoke external sequential debuggers that might implement some other architecture dependent debugging facilities on a specified host/pool in the heterogeneous environment. In this way the user can choose the best tool in every phases of debugging procedure. Additional tightly integrated graphical tools are responsible for the navigation through the distributed/Java virtual machines and threads (equipped by filtering options for handling of scalability), management of breakpoint sets and establishment of new debug sessions.

Finally, X-IDVS is *programmable* with a simple macro language particularly for testing purposes. Thus, the programmer can test the startup of his application and can force the metacomputing application to run with vary timing conditions.

3 Systematic debugging in Harness

As it was described above, X-IDVS was designed originally for Harness applications built on RMI-based plug-ins. During an RMI-based interaction the invoked remote methods are executed in separated threads on the server side but the macrostep debugging methodology [7] cannot be applied in case of multithreaded applications (which might use shared objects). Thus, we had to take into consideration two options: (i) attempt to extend the macrostep debugging methodology with multithreaded/shared objects support or (ii) provide systematic debugging support for other types of Harness plug-ins, e.g. which are based on message passing paradigm. As the first stage of this project, we applied the macrostep debugging methodology on Harness plug-ins which can communicate with each other via message box. Based on these experiences and achievements we will try to solve the systematic debugging issues of multithreaded/RMI-based metacomputing applications as the next stage of this project.

In Harness the message box plug-in provides a generic send/receive/scatter/gather message passing service for Harness plug-ins via a simple interface:

- public void *send*(String senderID, String destination, Object message)
- public void *sendToAny*(String senderID, Object message)
- public H_Envelope *receive*(String myID, String senderID)
- public H_Envelope *receiveFromAny*(String myID)
- public H_Envelope *receiveAsync*(String myID, String senderID)
- public H_Envelope *receiveFromAnyAsync*(String myID)

In details, the *send* and *sendToAny* operations are always executed asynchronously but each type of *receive* operation can be either asynchronous or synchronous. As a first step we reduced these communication possibilities in order to get a similar message passing interface as in P-GRADE system where the macrostep debugging methodology has been implemented for the first time. Thus, we turned the asynchronous send operations to synchronous send and also removed both asynchronous receive operations.

The main ideas of the further developed macrostep debugging methodology can be summarized by the following concepts: (i) enhanced collective breakpoints, (ii) modified macrosteps, (iii) extended macrostep-by-macrostep execution mode, (iv) execution tree, (v) meta-breakpoints, (vi) execution tree. In the rest of this section we describe these concepts as well as some implementation issues.

In [7], a restriction was introduced on the global breakpoint sets and introduced a special version of them called **collective breakpoints**. When all the breakpoints of the global breakpoint set are placed on communication instructions, the global breakpoint set is called collective breakpoint. A formal definition of the collective breakpoints can be found in [7]. If there is at least one breakpoint for each alternative execution path of every process, the collective breakpoint is called strongly complete. In

practice, we were able to implement the strongly complete collective breakpoints by placing breakpoints on each method entries of message box interface. It means only a couple permanent breakpoints for each message box thus, we might achieve good performance that can be crucial in case of communication intensive metacomputing programs. Two problems were turned out during the design phase: (i) RMI communication between plug-ins and the message box, (ii) dynamically created message boxes. In details; the message box service was implemented as a plug-in, according to the Harness concept, and the sender/receiver plug-ins have to communicate with the message box plug-in *via RMI*. As it described in [15] JPDA has no debugging support for RMI but we have to find out which plug-in wants to send or receive a message (the *myID* and *senderID* string arguments can be defined without any restrictions by plug-ins). Hereby, we had to deal with issues of RMI debugging and to apply some RMI-related functions of X-IDVS in spite of our original plans. On the other hand, any Harness plug-in can create dynamically *new message boxes* therefore; our debugger tool must be also responsible for detecting when a new message box plug-in is loaded.

The set of executed code regions between two consecutive collective breakpoints is called a **macrostep**. Precise definition of macrostep is given in [7]. Provided that sequential program parts between communication instructions are already tested, a systematic debugging of a metacomputing program requires to debug the metacomputing program by pure macrosteps, i.e. instrumenting all the communication instructions by global breakpoints. A breakpoint of the collective breakpoint is called active if it was hit in a macrostep and its associated instruction has been completed. A breakpoint is called sleeping if it was hit in a macrostep but its associated instruction has not been completed (for example, receive instruction waiting for a message). Those breakpoints that were either active or sleeping in a macrostep are together called effective breakpoints.

After the definitions given above we can define the **macrostep-by-macrostep execution mode** of metacomputing programs. In each step either the user or the debugger runs the program until the collective breakpoint is hit. Under these conditions the metacomputing program will be executed by macrostep-by-macrostep. The boundaries of the macrosteps are defined by a series of effective global breakpoint sets. In such cases the user is interested only in checking the program state at the well-defined boundary conditions.

There is a clear analogy between the step-by-step execution mode of sequential programs realised by local breakpoints and the macrostep-by-macrostep execution mode of metacomputing programs. The macrostep-by-macrostep execution mode enables to check the progress of the metacomputing program at the points that are relevant from the point of view of parallel execution, i.e. at the message passing points. What we should ensure is that the macrostep-by-macrostep execution mode should work deterministically just like the step-by-step execution mode works in case of sequential programs. In order to ensure it, according to the original macrostep concept the debugger should store the history of collective breakpoints, the acceptance order of messages at receive instructions and the result of input operations. Additionally, in a metacomputing environment we should also store the events about the reconfiguration; when a new plug-in is loaded, unloaded or failed anywhere in heterogeneous computational environment, new host is grabbed/released or a new

message box is started by the user's application. Therefore, our debugger tool must be able to adapt to the dynamic behaviour of debugged application as well as its fault tolerance. As it was mentioned in Section 2.1, the enrolled computational resources as well as the DVM itself can be reconfigured. To handle the dynamic, reconfigurable and fault tolerant behaviour of DVM, our basic idea was the following. During the initialisation the Harness Monitor/Control Plug-In (HMCPI) places some so-called 'system breakpoints' in the Harness kernel (see Figure 2) in order to detect the changes/reconfiguration of DVM in advance. Then, HMCPI can report these events to Harness Systematic Debugger Tool (HSDT) that is responsible for storing these reconfiguration events in a trace file (see Figure 2). Basically, the fault tolerance of X-IDVS has been inherited from the Harness Framework itself.

At replay, the progress of tasks are controlled by the stored collective breakpoints and reconfiguration events and the program is automatically executed again macrostep-by-macrostep as in the execution phase. The debugger is also responsible for loading/unloading/killing the plugins, grabbing/releasing hosts and starting new message boxes during each macrostep (if it is needed). Obviously, during the replay phase it is not guaranteed that a host can be grabbed again for the distributed virtual machine or a given host is able to load the required plug-in (resource limitations, etc.). Our solution is a *host translation table* maintained by the debugger, in that each host enrolled in the original DVM can be associated to a substitute host (independently for each plug-in) where the appropriate plug-in actually run during the replay phase. The relative speed of the substitute host is unessential because the macrostep-by-macrostep execution can handle this issue. Only the architecture of the substitute host can be important if the current plug-in uses some architecture dependent features (e.g. via Java Native Interface). In this case, we have to check whether both architectures of reference and substitute hosts are the same ones.

In Harness the introduced host-translation table is used by the systematic debugger tool as well as the RMI communication core during the replay/control phases.

The execution path is a graph whose nodes represent macrosteps and the directed arcs connect the consecutive macrosteps. The **execution tree** is a generalization of the execution path, it contains all the possible execution paths of a metacomputing program assuming that the non-determinism of the current program is inherited from (wildcard) message passing communications. Nodes of the execution tree can be of four types: (i) Root node, (ii) Alternative nodes, (iii) Deterministic nodes.

The Root node represents the starting conditions of the metacomputing program. Alternative nodes indicate either a wildcard receive instructions which can choose a message non-deterministically from several processes or (as an extension of the original macrostep concept) a wildcard send instructions which can send a message non-deterministically to any process. Only alternative nodes can create new execution paths in the execution tree, deterministic nodes cannot create any new execution path.

Breakpoints can be placed at the nodes of the execution tree. Such breakpoints are called **meta-breakpoints**. The role of meta-breakpoints is analogous with the role of the breakpoints of sequential programs. A breakpoint in a sequential program means to run the program until the breakpoint is hit. Similarly, a meta-breakpoint at a node of the execution tree means to place the collective breakpoint belonging to that node and run the metacomputing application until the collective breakpoint is hit. Replay

guarantees that the collective breakpoint will be hit and the metacomputing program will be stopped at the requested node.

The task of systematic debugging or testing is to exhaustively traverse the complete execution tree with all the possible execution paths in it. Therefore, the execution tree represents a search space that should be completely explored by the debugging method. Accordingly, systematic testing and debugging of a metacomputing program require (i) generation of its execution tree (ii) exhaustive traverse of its execution tree. With the help of the extended macrostep-by-macrostep concept both of these issues can be solved and implemented in a very similar way as they have been implemented in DIWIDE [8][12]. Some minor changes are required by the wildcard send operations as well as the event tracing and replaying.

Often a Harness plug-in does not require a particular architecture for its execution. Despite of this we always have to inspect whether each plug-in has been implemented *architecture independently* if we want to get a real metacomputing application. For testing the architecture independency of plug-ins or whole applications the *systematically* generated host-translation tables are needed. It means that we have to test each architecture independent plug-in on each significantly different architecture (by exhaustive traverse of the execution tree).

We can test several plug-ins (from the aspect of architecture dependency) in one exhaustive traverse of an execution tree. In the best case we need only as much traversing of the execution tree as the number of significantly different architecture we have in the metacomputing environment. Our solution contains four steps: (1) the debugger looks for a host with a new and untested architecture for each architecture independent and not fully tested plug-in and the debugger also registers the found host into the new host-translation table, (2) if step 1 was not successful by any plug-in (after a timeout), we allowed the debugger to look for *any* host for the unsuccessfully mapped plug-ins (3) if there was at least one successfully mapped plug-in among the not fully tested plug-ins the debugger starts exhaustive traverse of execution tree, else go to Step 1, (4) if there is any not fully tested plug-in, go to Step 1.

We can decrease the time and resource requirements of the exhaustive tests by magnitude of orders in two ways. On one hand, we can take the advantage the large number of resources included in metacomputing environment by starting more (hundreds or thousands) test scenarios at the same time. On the other hand, we can try to reduce the complexity/size of metacomputing application as much as possible without losing the relevant parts of the application.

4 The architecture of systematic debugger

First of all, Harness kernels (see Figure 2) are launched with a special debug flag in order to enable the JVMDI interfaces and turn the JIT compiler off. As depicts in Figure 2, on each host enrolled in the distributed virtual machine two different types of plug-in can be found for debugging purposes: one HDPI and one HMCPI plug-in. Both of them are loaded by HSDT (with the same authorization keys as the user plug-ins) during the initialisation phase but they play different roles. In the first step of initialisation, HDPI gathers information about the enabled JVMDI interface and pass

the information to the HSDT. Therefore, in the second step the HSDT is able to load the HMCPI plug-in and attach it to the JVM of Harness kernel with the help of JDWP. Then, HMCPI is able to (i) place the system breakpoints in the Harness kernel in order to detect the reconfiguration of DVM, (ii) monitor the access the message boxes, (iii) control the execution of plug-ins.

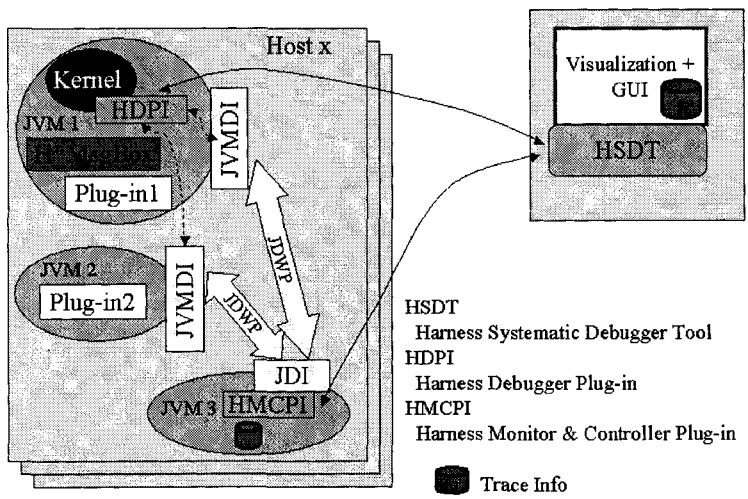


Fig. 2. Fundamental Architecture of Systematic Debugger In Harness

5 Conclusion and related work

As we studied the related work (such as DejaVu [1], DJVM [11], DIWIDE [8][12] integrated in P-GRADE [10], Macrostep-by-macrostep concept [7], TotalView, P2D2 [4]) we realized that there is a lack of an integrated graphical systematic debugger for metacomputing applications equipped by visualization techniques. Even the most relevant grid/metacomputing projects such as Globus [3], Condor [13] and Legion [6] do not give efficient solution to the emerging debugging issues. That was the main reason for the further development of the macrostep debugging methodology. The current prototype is partly implemented under the Harness Framework as an extension of X-IDVS metadebugger.

We also plan to investigate the efficiency issues of the described methodology as well as the feasible optimisation techniques that could be applied in order to reduce the complexity of exhaustive testing.

6 References

[1] Bowen Alpern, Jong-Deok Choi, Ton Ngo, Manu Sridharan, John Vlissides. A Perturbation-Free Replay Platform for Cross-Optimized Multithreaded Applications. IBM Research Report, RC 21864, 22 September 2000.

[2] Jong-Deok Choi and Harini Srinivasan. Deterministic Replay of Java Multithreaded Applications. ACM SIGMETRICS Symposium on Parallel and Distributed Tools, pages 48-59, August 1998.

[3] I. Foster and C. Kesselman. Globus: a Metacomputing Infrastructure Toolkit. International Journal of Supercomputing Application, May 1997.

[4] Robert Hood. The p2d2 project: building a portable distributed debugger. Proceedings of the SIGMETRICS symposium on Parallel and distributed tools, May 22 - 23, 1996, Philadelphia, PA USA

[5] A. Geist, A. Beguelin, J. Dongarra, W. Jiang, B. Mancheck and V. Sunderam. PVM: Parallel Virtual Machine a User's Guide and Tutorial for Networked Parallel Computing, MIT Press, Cambridge, MA, 1994.

[6] A. Grimshaw, W. Wulf, J. French, A. Weaver and P. Reynolds. Legion: the next logical step toward a nationwide virtual computer, Technical Report CS-94-21, University of Virginia, 1994.

[7] P. Kacsuk. Systematic Macrostep Debugging of Message Passing Parallel Programs. Future Generation Computer Systems, Vol. 16, No. 6, pp. 609-624, 2000.

[8] P. Kacsuk, R. Lovas, J. Kovacs. Systematic Debugging of Parallel Programs in DIWIDE Based on Collective Breakpoints and Macrosteps. In: Proceedings. 5th International Euro-Par Conference, Toulouse, France, 1999. pp. 90-97.

[9] P. Kacsuk, G. Dózsa, T. Fadgyas, R. Lovas. GRADE: A Graphical Programming Environment for Multicomputers. Computer and Artificial Intelligence. 17 (5) :417-427. (1998)

[10] P. Kacsuk. Visual Message Passing Programming – the P-GRADE Concept. Scientific Programming Journal. 2000, Special Issue on SGI'2000

[11] Ravi Konuru, Harini Srinivasan, and Jong-Deok Choi. Deterministic Replay of Distributed Java Applications. 14th International Parallel & Distributed Processing Symphosium, pages 219-228, May 2000.

[12] J. Kovacs, P. Kacsuk. The DIWIDE Distributed Debugger on Windows NT and UNIX Platforms, Distributed and Parallel Systems, From Instruction Parallelism to Cluster Computing, Eds.: P. Kacsuk and G. Kotsis, Cluwer Academic Publishers, 2000.

[13] M. J. Litzkow, M. Livny and M. W. Mutka. Condor – A Hunter of Idle Workstations, Proc. of the 8th International Conference on Distributed Computer Systems, pp. 104-111, IEEE Press, June 1998.

[14] M. Migliardi, V. Sunderam, A.Geist, J. Dongarra. Dynamic Reconfiguration and Virtual Machine Management in the Harness Metacomputing System, Proc. of ISCOPE98, pp. 127-134, Santa Fe', New Mexico (USA), December 8-11, 1998.

[15] R. Lovas, V. Sunderam: Extendible Integrated Debugging and Visualization Service for Harness Metacomputing Framework, technical paper, available online at: http://www.sztaki.hu/~rlovas/projects/harness/docs/xidvs.pdf

Capacity and Capability Computing Using Legion*

Anand Natrajan, Marty A. Humphrey, and Andrew S. Grimshaw

Department of Computer Science, University of Virginia, Charlottesville, VA 22904

{anand, humphrey, grimshaw}@cs.virginia.edu

Abstract. Computational Scientists often cannot easily access the large amounts of resources their applications require. *Legion* is a collection of software services that facilitate the secure and easy use of local and non-local resources by providing the illusion of a single virtual machine from heterogeneous, geographically-distributed resources. This paper describes the newest additions to Legion that enable high-performance (capacity) computing as well as secure, fault-tolerant and collaborative (capability) computing.

1 Introduction

As available computing power increases because of faster commodity processors and faster networking, computational scientists are attempting to solve problems that were considered infeasible until recently. However, merely connecting large machines with high-speed networks is not enough; an easy-to-use and unified software environment in which to develop, test and conduct software experiments is absent. For example, users often are forced to remember multiple passwords, copy files to and from machines, determine where necessary compilers and libraries are on each machine, and choose which machines to use at particular times.

A *metasystem* is an environment in which users, such as scientists, can access resources in a transparent and secure manner. In a metasystem, users are not limited by geography, by non-possession of accounts, by limits of resources at one site or another and so on. In short, as long as a resource provider is willing to permit a user to use the resource, there is no barrier between the user and the resource.

Legion is an architecture for a metasystem [1]. Just as an operating system provides an abstraction of a machine, Legion provides an abstraction of the

* This work was supported in part by the National Science Foundation grant EIA-9974968, DoD/Logicon contract 979103 (DAHC94-96-C-0008), and by the NASA Information Power Grid program.

V.N. Alexandrov et al. (Eds.): ICCS 2001, LNCS 2073, pp. 273–283, 2001.
© Springer-Verlag Berlin Heidelberg 2001

metasystem. This abstraction supports the current performance demands of scientific applications. A number of scientific applications already run using Legion as the underlying infrastructure. In the future, scientists will demand support for new methods of collaboration. Legion supports these expected demands as well.

We define *capacity computing* loosely as the ability to conduct larger computational experiments either by expending more resources on a single problem or on multiple, independent problems. We define *capability computing* to be new mechanisms with which to conduct computational science experiments. This paper describes, from the viewpoint of a computational scientist, Legion's unique support for high-performance capacity and capability computing and describes how computational scientists in a variety of disciplines are using Legion today.

2 Legion

The Legion project is an architecture for designing and building system services that present users the illusion of a single virtual machine [2]. This virtual machine provides secure shared objects and shared name spaces. Whereas a conventional operating system provides an abstraction of a single computer, Legion aggregates a large number of diverse computers running different operating systems into a single abstraction. As part of this abstraction, Legion provides mechanisms to couple diverse applications and diverse resources, vastly simplifying the task of writing applications in heterogeneous distributed systems.

Each system and application component in Legion is an object. The object-based architecture enables modularity, data and fault encapsulation and replaceability — the ability to change implementations of any component. Legion provides persistent storage, process management, inter-process communication, security and resource management services, long regarded as the basic services any operating system must provide. Legion provides these services in an integrated environment, not as disjoint mechanisms such as Globus does [3]. Of particular importance is the integration of security into Legion from the design through implementation. Legion supports PVM [4], MPI [5], C, Fortran (with an object-based parallel dialect), a parallel C++ [6], Java and the CORBA IDL [7]. Also, Legion addresses critical issues such as flexibility and extensibility, site autonomy, binary management and limited forms of fault detection/ recovery. From inception Legion was designed to manage millions of hosts and billions of objects — a capability lacking in other object-based distributed systems [8].

3 Capacity Computing with Legion

Legion can benefit scientific applications by delivering large amounts of resources such as computing power, storage space and memory. Moreover, Legion provides a rich set of tools that make the access and use of these resources simple and straightforward. In particular, there are tools for running programs written using MPI and PVM as well as programs that are parameter-space studies or sequential

codes. In §3.1-§3.4, we present some of Legion's tools for running applications. In §3.5, we discuss scheduling in Legion briefly.

3.1 Legacy Applications

Legacy applications are those whose source code does not consist of any calls to Legion routines and does not utilise Legion objects and tools. Moreover, the source code of the application may not be modified to target it to Legion, either because it is unavailable or because its authors are unavailable or unwilling to make the necessary changes. In all such cases, Legion neither mandates retargetting the application nor denies access to metasystem resources.

A Legion user may run a legacy application on the distributed resources of a metasystem by undertaking two steps (tool names are in parentheses):

1. Register the executable as a runnable class (`legion_register_program`)
2. Run the class (`legion_run`)

The first step results in the creation of a *runnable class*, analogous to an executable in Unix or Windows. Registering an executable is an infrequent step, required only when the runnable class does not exist in Legion or when the executable available to the user changes. A user is likely to execute the second step repeatedly in order to initiate, monitor and complete repeated runs of the application. The executable registered with this class is called an *implementation*. Multiple executables, typically of different architectures, may be registered with the same class, for example:

```
legion_register_program myClass /bin/whoami solaris
legion_register_program myClass /bin/ls sgi
```

The first command creates a class `myClass`, which Legion tools can recognise as a runnable class. The second parameter to the command is the Unix (or Windows) path of an executable to be registered as an implementation for `myClass`. The third argument indicates that the executable is a Solaris binary. When the second command is executed, Legion recognises that `myClass` exists. It adds the binary `/bin/ls` as an SGI implementation for the same class. Subsequently, if a user runs `myClass` on a Solaris machine, the executable corresponding to `/bin/whoami` will be executed on that machine, whereas if the user runs `myClass` on an SGI machine, the executable corresponding to `/bin/ls` will be executed on that machine.

This example is trivial in the sense that `/bin/whoami` and `/bin/ls` are not high- performance applications. Moreover, running `myClass` on different architectures is likely to give very different results. However, the example illustrates that (a) registering legacy applications in Legion is simple and (b) no semantic requirement is imposed on the executables registered for different architectures.

Once a runnable class has been created in Legion, a user can run the class by issuing a `legion_run` command. The simplest form of the command is:

```
legion_run myClass
```

Here, the user implies that Legion can run an instance of the class on any resource present in Legion provided (a) `myClass` has implementations for the machine on which the instance eventually runs (i.e., Solaris or SGI implementations), (b) the user is permitted to run on the machine and (c) the machine accepts the instance for running. A more sophisticated run is:

```
legion_run -v -IN file1 -OUT file2 myClass convert
```

Here, the user indicates that she will observe the run in verbose mode (`-v`), will provide one Unix or Windows input file (`-IN file1`) and will receive one Unix or Windows output file (`-OUT file2`) when running an instance of `myClass` with the argument `convert`. Legion ensures that the input and output files are copied to and from the machine on which the instance runs. In this form as well, the user has indicated that she prefers Legion to select the machine on which the instance runs. While this transparency in scheduling is used often, some users happen to be aware of the machine on which they would like to run. Therefore, Legion permits directed scheduling, wherein the user specifies the machine on which she wants to run:

```
legion_run -h /hosts/xyz -IN file1 -OUT file2 myClass convert
```

The details of how individual runs can be configured to suit a user's requirements are beyond the scope of this paper. It suffices to say that in keeping with the Legion philosophy of providing mechanisms on which policies can be constructed, there exist many different strategies for executing a legacy application on distributed resources. These different strategies can be applied by choosing from a large number of options available in `legion_run`. The options are part of the standard documentation and man pages available at each Legion installation [9].

3.2 MPI Applications

Many high-performance parallel applications are written using the Message Passing Interface (MPI) library [5]. An MPI library provides routines that enable communication among various processes of a parallel application. MPI is a standard, i.e., it defines the interface of the routines. Different vendors of MPI may implement a routine differently provided they adhere to the standard interface. Legion's support for MPI is three-fold: Legion MPI, native MPI and mixed MPI.

Legion MPI. Legion can be viewed as another MPI vendor because it provides implementations to standard MPI routines. If a user desires that an application using MPI routines should run on a metasystem, he has to undertake three simple steps:

1. Re-link the object code of the application with Legion libraries (`legion_link`)
2. Register the executable as an MPI runnable class (`legion_mpi_register`)
3. Run the class (`legion_mpi_run`)

The first step ensures that Legion's implementation of MPI routines are used when running the application. Note that it is not necessary to change the source

code of the application. The subsequent steps are similar to those for legacy applications. The options and operations of the actual commands are similar to those for registering and running legacy applications.

Native MPI. Some MPI applications are intolerant of high latencies for inter- process communications. Running such applications on distributed resources may degrade the performance of the application. Such applications are better supported by running them on proximal resources to reduce communications latency. Moreover, many MPI implementations are tuned finely to exploit the architecture of underlying resources. Finally, the users of many MPI applications may be unwilling or unable to re-link the application with Legion libraries. Therefore, Legion supports running MPI applications in "native" mode, i.e., using other implementations of MPI, such as MPICH [10]. Native MPI support is similar to support for Legion MPI as well as legacy applications. The steps a user has to undertake are:

1. Register the executable as a runnable class (`legion_native_mpi_register`)
2. Run the class (`legion_native_mpi_run`)

The benefits to the user are that no recompiling or re-linking is necessary to access remote resources in a transparent manner.

Mixed MPI. Mixed MPI support is a blend of Legion MPI and native MPI. In Legion's mixed MPI support, an application is executed in "native" mode, but the application can access Legion's objects, such as files. The steps required are:

1. Modify source code to initialize Legion library
2. Re-link the object code of the application with Legion libraries (`legion_link`)
3. Register the executable as a runnable class (`legion_native_mpi_register`)
4. Run the class (`legion_native_mpi_run -legion`)

The user has to modify the source code to initialize Legion with one call from within the application. Registering and running the class is similar to native MPI with the addition of one option. Applications written to take advantage of mixed MPI support can benefit in two ways: (a) since runs are executed in native mode, performance for latency-intolerant applications does not suffer, and (b) runs can access Legion objects and thus take advantage of the metasystem.

3.3 Mentat and Basic Fortran Support (BFS)

High-performance applications can be supported in Legion if they are written in Mentat or if they use the Basic Fortran Support. Mentat is a language similar to C++ with a few additional keywords [6]. In Mentat, users may specify classes to be stateless or persistent. The Mentat compiler identifies data dependencies within a program and constructs a dataflow graph to execute the program. Mentat provides a platform for users to write high-performance applications using a

compiler constructed to mask the tedium of writing parallel programs. Legion's support for Fortran programs is called BFS [11]. If users desire to write metasystem applications in Fortran, then Legion requires that metasystem directives be embedded within Fortran comments. Currently, BFS support targets Mentat, but may not in future releases.

3.4 Parameter-Space Studies

Many metasystem applications are parameter-space (p-space) studies. In a p-space study, a single program is called repeatedly with different sets of parameters. Multiple instances of the program may run concurrently with different sets of parameters. These instances are completely independent of one another. Therefore, they can be scheduled easily across geographically-distributed resources.

With Legion's support, users may run their p-space studies orders of magnitude faster than sequential. First, the application must be registered (see §3.1-§3.2). Next, the user must indicate which files must be mapped to the files required by an instance. Finally, the application must be run with `legion_run_multi`. Legion runs each instance of the application by mapping the proper files for the instance and copying output files appropriately. `legion_run_multi` takes a number of options in order to tailor the running of a p-space application for a user. This tool ensures that input files and output files are arranged such that the user can identify corresponding sets easily.

3.5 Scheduling

In a metasystem, scheduling is the process of initiating runs on the best possible resources. The general scheduling problem is NP-complete [12]. In addition, the parameters involved in making an optimal schedule are numerous and mutually dependent. Constructing a schedule may involve making decisions not limited to: (a) the machine architectures for which a class has implementations, (b) specific properties of a machine desired by the class (e.g., is it a queuing system? can it run MPI jobs natively?), (c) communication bandwidth *versus* performance penalty, (d) current load and storage space on the machine, (e) permissions for this user to run an instance of this class on that machine, (f) allocation remaining for the user on that machine and (g) charges imposed by resource providers for running on their machine.

Legion provides mechanisms to construct schedulers. Different schedulers may employ different algorithms to construct schedules from the list of available resources. Also, Legion permits users to specify resources directly for a run, the rationale being that until good heuristics are developed to address all issues in scheduling, users are likely to be the best schedulers of their own jobs.

The general scheduling architecture in Legion is based on negotiation between resource providers and consumers [13]. The negotiation process preserves autonomy of resource providers while satisfying the demands of the consumers. When a user starts a run, Legion encapsulates the demands of the user in the

run request. The scheduler uses this request to construct one or more schedules for this run. Next, it queries the resource objects in turn to determine if they will accept the run. The resource objects may exercise the autonomy of the resource providers in accepting or denying the run. If they accept, the runs are initiated on the chosen resources.

4 Capability Computing with Legion

A well-designed metasystem should not only satisfy current demands of users but also anticipate and satisfy future demands. Currently, many applications require high performance. However, in the near future, metasystems such as Legion will be able to deliver high performance to applications routinely by providing access to distributed resources. We believe that at that point, users will look beyond high performance as the defining feature of a metasystem. At that point, users' demands may include heterogeneity, security, fault-tolerance and collaboration.

Heterogeneity is a fundamental design principle in Legion [14]. Typically, a running metasystem that uses Legion incorporates diverse resources — machines of different architectures running different operating systems consisting of different configurations and managed by different organisations. As in §3, users may register implementations of different architectures for their runnable classes. For parallel applications, different instances started by a single run may run on heterogeneous machines and communicate with one another as if they ran on homogeneous machines.

Security was designed into Legion from the start [15]. Every Legion object, whether it be a resource, a user, a file, a runnable class or a running instance, has a security mechanism associated with it. The mechanisms provided by Legion are general enough to accommodate different kinds of security policies within a single metasystem. Typically, the security provided is in the form of access control lists. An access control list indicates which objects can call which methods of an object. This fine-grained control mechanism enables users and metasystem administrators to set sophisticated policies for different objects. The authentication mechanism currently employed by Legion is a public key infrastructure based on key pairs. The keys are used to encrypt and decrypt messages securely as well as for signing certificates.

Fault-tolerance can be implemented in a number of ways in Legion [16]. Basic Legion objects are fault-tolerant because they can be deactivated at any time. When a Legion object is deactivated, it saves its state to persistent storage and frees memory and process state. Subsequently, it may be reactivated from its persistent state either on the same or a different machine. If it is reactivated on a different machine, Legion transfers its state to the new machine whenever possible. In addition, some objects can be replicated for performance or availability. Legion's MPI implementation provides mechanisms for checkpointing, stopping and restarting individual instances. Finally, Legion provides tools for retrieving intermediate files generated by legacy applications. Users can restart their instances using these intermediate files.

Legion enables new paradigms for collaboration between researchers conducting experiments that require using metasystem resources. We believe that collaboration is an important goal for a metasystem. We expect that researchers should not be limited by geographical distance between one another as well as the resources they desire to use. Accordingly, the ability to share objects via their permissions (access control lists) has always been a key design feature in Legion. In §4.1 and §4.2, we outline some of the methods by which users of a metasystem can collaborate.

4.1 Context Space

Legion provides a shared, virtual space to metasystem users. The shared, virtual space can be viewed as a truly distributed, global file system. This file system is organised in a manner similar to a Unix file system. In order to distinguish the global file system from the file systems present on individual machines, we call the global file system a *context space*. Directories in context space are called *contexts*. A context called "/" typically denotes the root of the context space. A context is an object that contains other objects — contexts, machines, users, classes, files, etc. All users of a metasystem, no matter where located physically, have the same view of the context space. The analogue of this model in traditional operating systems is an NFS-mounted disk that is visible to all machines that share the mount, or a Samba-mounted Unix directory that is visible from a Windows machine.

The scope of Legion's context space is much vaster than that of any of its predecessors. Distributed file systems are not novel. Legion's implementation has predecessors in Network File System (NFS) [17], the Andrew File System (AFS) [18] and Extensible File System (ELFS) [19]. However, context space is truly distributed and global; individual components may be physically located on machines that do not have anything in common except that they are part of the same metasystem.

Users may freely transfer files from their local file systems to context space. For example, one of the options to a tool called `legion_cp` permits users to copy a text file from their file system to context space. Likewise, registering a program effectively transfers an executable from a local file system to context space. A growing number of tools available in Legion permit users to interface with context space in novel ways. For example, a tool called `legion_export_dir` lets a user mirror an entire directory in his local file system into Legion. Likewise, a Windows tool lets users browse context space. When these two tools are used in conjunction, a user on one Windows machine may be able to view the contents of his collaborator's directories on another Windows machine across the globe. Naturally, the permissions on the exported directory and its components have to be set to permit the collaborator (and perhaps only the collaborator) to view them. However, setting the permissions is a matter of manipulating the access control lists of the objects. Legion provides tools for manipulating the access control lists of objects.

Tools for traversing context space include a suite of Unix-like command-line tools, a point-and-click Web browser interface, an FTP tool, a Samba interface for Windows, an HTTP interface, and a Legion implementation of NFS for accessing context space with standard Unix tools such as `ls` and `cat` as well as with standard system calls like `open`, `read` and `write` [20]. Using these tools, metasystem users can collaborate by sharing and exchanging data in a manner familiar to them. Moreover, because of the possibility of setting fine-grained access controls, collaborators can also select the level of collaboration.

4.2 Sharing Runs

Legion's object model is flexible enough to permit novel means of collaboration among researchers, for example, sharing runs. In Legion, running instances of a class are first-class objects themselves. Therefore, as with any object in Legion, access control lists can be set for them to control permissions in interesting ways.

Suppose two researchers situated across a country wish to collaborate. The nature of their collaboration requires one of them to initiate a run which both observe. Currently, such a collaboration would be impossible unless both researchers were able to share an account on some machine. In Legion, neither researcher would need an account on the machine on which the instance runs. Instead, both could access the same object using Legion tools from their own machines.

Suppose a researcher constructs an application that is likely to be used widely by others in the same field. The researcher could register her executable as a runnable class in Legion and set the permissions to allow anyone, a group of users or an *a priori* known set of users to run instances of the class. Currently, the researcher would have to send or sell her executable to her fellow researchers. In the Legion model, she could control who runs her class when, where and how many times without physically transporting her executable to the other researchers' machines.

Suppose two mutually distrustful parties wish to collaborate on an experiment with one providing the executable and the other the data. Currently, such a collaboration is impossible because either the executable or the data must be transported to the other collaborator. However, in Legion, such a collaboration is legitimate and possible. The collaborator with the executable would register the executable as a class in Legion and start an instance. Then he would set the permissions on the instance allowing only the other collaborator to perform data transfers but retaining permission to terminate the experiment. The second collaborator, after verifying that the permissions are indeed as outlined above, could commence transferring data files. The application in question would have to be written in such a manner that it can wait until the data files become present. With that minor change in place, Legion can enable these mutually distrustful parties to collaborate.

Other means of collaboration will become evident as metasystems are used more widely and routinely. We expect the Legion model to be flexible enough to accommodate these collaboration efforts as they arise.

5 Conclusions and Current Status

The success of a metasystem depends on how easily and securely it permits users to perform their computations by collaborating and accessing available resources. A key component of a metasystem is software that presents users with abstractions of resources. Legion provides those abstractions *via* uniform, easy-to-use interfaces. These interfaces, ranging from tool-level to programming-level, greatly reduce the difficulties of computing in distributed, heterogeneous environments. The mechanisms underlying the interfaces enable users to perform cross-machine, cross-architecture and cross-organisation computation. By enabling such computations on a large scale, Legion supports capacity computing. Legion's flexible and extensible object model supports capability computing by permitting novel methods of computation.

Legion consists of 350,000 lines of code and has been ported to Windows NT as well as a large number of Unix variants, including Linux (Intel, Alpha), Unicos (T90, T3E), AIX (SP-2, SP-3), HPUX, FreeBSD, IRIX (Origin 2000) and Solaris (Enterprise 10000). Legion has been integrated with a large number of queuing systems, such as PBS, LSF, Codine, LoadLeveler and NQS. It has been deployed on machines belonging to NSF-PACI, NASA IPG and the DoD MSRCs. Currently, Legion is running at over 300 hosts across the United States and Europe. Researchers using Legion currently are from a number of disciplines, including:

- Biochemistry (e.g., complib, a protein and DNA sequence comparison)
- Molecular Biology (e.g., CHARMM, a p-space study of 3D structures)
- Materials Science (e.g., DSMC, a Monte Carlo particle-in-cell study)
- Aerospace (e.g., flapper, a p-space study of a vehicle with flapping wings)
- Information Retrieval (e.g., PIE, a personalised search environment)
- Climate Modelling (e.g., BT-MED, a 2D barotropic ocean model)
- Astronomy (e.g., Hydro, a study of a rotating gas disk around a black hole)
- Neuroscience (e.g., a biological-scale simulation of a mammalian neural net)
- Computer Graphics (e.g., a parallel rendering of independent movie frames)

We expect users to become more accustomed to using distributed resources, often in ways not anticipated today. Legion's architecture holds the promise of satisfying the computational demands of the present as well as the future.

References

1. Grimshaw, A. S., Ferrari, A. J., Lindahl, G., Holcomb, K., Metasystems, Communications of the ACM, Vol. 41, No. 11, November 1998.
2. Grimshaw, A. S., Wulf, W. A., The Legion Vision of a Worldwide Virtual Computer, Communications of the ACM, Vol. 40, No. 1, January 1997.
3. Foster, I., Kesselman, C., The Grid: Blueprint for a New Computing Infrastructure, Morgan Kaufmann, 1999.

4. Geist, A., Beguelin, A., Dongarra, J., Jiang, W., Manchek, R., Sunderam, V., PVM: Parallel Virtual Machine: A User's Guide and Tutorial for Networked Parallel Computing, MIT Press, 1998.
5. Snir, M., Otto, S., Huss-Lederman, S., Walker, D., Dongarra, J., MPI: The Complete Reference, MIT Press, 1998.
6. Grimshaw, A. S., Ferrari, A. J., West, E., Mentat, Parallel Programming Using C++, MIT Press, 1996.
7. Seigel, J., CORBA Fundamentals and Programming, Wiley, ISBN: 0471-12148-7, 1996.
8. Grimshaw, A. S., Ferrari, A. J., Knabe, F., Humphrey, M. A., Wide-Area Computing: Resource Sharing on a Large Scale, IEEE Computer, Vol. 32, No. 5, May 1999.
9. —, The Legion Manuals (v1.7), University of Virginia, October 2000.
10. Gropp, W., Lusk, E., Doss, N., Skjellum, A., A High-Performance, Portable Implementation of the Message Passing Interface Standard, Parallel Computing, Vol. 22, No. 6, September 1996.
11. Ferrari, A. J., Grimshaw, A. S., Basic Fortran Support in Legion, University of Virginia Technical Report CS-98-11, March 1998.
12. Weissman, J., Scheduling Parallel Computations in a Heterogeneous Environment, Ph.D. Dissertation, University of Virginia, August 1995.
13. Chapin, S. J., Katramatos, D., Karpovich, J. F., Grimshaw, A. S., Resource Management in Legion, University of Virginia Technical Report CS-98-09, February 1998.
14. Grimshaw, A. S., Lewis, M. J., Ferrari, A. J., Karpovich, J. F., Architectural Support for Extensibility and Autonomy in Wide-Area Distributed Object Systems, University of Virginia Technical Report CS-98-12, June 1998.
15. Ferrari, A. J., Knabe, F., Humphrey, M. A., Chapin, S. J., Grimshaw, A. S., A Flexible Security System for Metacomputing Environments, High Performance Computing and Networking Europe, April 1999.
16. Nguyen-Tuong, A., Integrating Fault-tolerance Techniques in Grid Applications, Ph.D. Dissertation, University of Virginia, August 2000.
17. Sandberg, R., Goldberg, D., Kleiman, S., Walsh, D., Lyon, B., Design and Implementation of the SUN Network File System, Proceedings of USENIX Conference, 1985.
18. Howard, J., Kazar, M., Menees, S., Nichols, D., Satyanarayanan, M., Sidebotham, R., West, M., Scale and Performance in a Distributed File System, ACM Transactions on Computer Systems, Vol. 6, No. 1, February 1988.
19. Karpovich, J. F., Grimshaw, A. S., French, J. C., Extensible File Systems (ELFS): An Object-Oriented Approach to High Performance File I/O, 9_th Annual Conference on Object-Oriented Programming Systems, Languages and Applications (OOPSLA), October 1994.
20. White, B. S., Grimshaw, A. S., Nguyen-Tuong, A., Grid-Based File Access: The Legion I/O Model, High Performance Distributed Computing 9, August 2000.

Component Object Based Single System Image Middleware for Metacomputer Implementation of Genetic Programming on Clusters

Ivan Tanev[1], Takashi Uozumi[1], and Dauren Akhmetov[2]

[1] Department of Computer Science and System Engineering, Muroran Institute of Technology, Mizumoto 27-1, Muroran 050-8585, Japan
i.tanev@computer.org, uozumi@csse@muroran-it.ac.jp

[2] Solution Department, Mobile Multimedia Business Headquarters, NTT DoCoMo Hokkaido, Inc, 6-Nishi-1, West-14, Chuo-ku, Sapporo 060-0001, Japan
dauren_akhmetov@nttdocomo-h.co.jp

Abstract. We present a distributed component-object model (DCOM) based single system image middleware (SSIM) for metacomputer implementation of genetic programming (MIGP). MIGP is aimed to significantly improve the computational performance of genetic programming (GP) exploiting the inherent parallelism in GP among the evaluation of individuals. It runs on cost-effective clusters of commodity, non-dedicated, heterogeneous workstations. Developed SSIM represents these workstations as a unified virtual resource and addresses the issues of locating and allocating the physical resources, communicating between the entities of MIGP, scheduling and load balance. Adopting DCOM as a communicating paradigm offers the benefits of software platform- and network protocol neutrality of proposed implementation; and the generic support for the issues of locating, allocating and security of the distributed entities of MIGP. Presented results of experimentally obtained speedup characteristics show close to linear speedup of MIGP for solving the time series identification problem on cluster of 10 W2K workstations.

1 Introduction

Genetic programming (GP) [1] is an algorithmic paradigm, inspired by the natural evolution of species. GP is successfully applied for solving increasingly difficult problems in artificial intelligence such as electrical circuits design, evolving digital hardware, spatial information classification, time-series identification, etc. [1]. However, GP is computationally costly - running even moderately sized problems, to which GP is applied, often requires hours on currently available computer architectures. One of the ways of speeding-up the GP is to improve the computational performance of the implementation. One of the most promising approaches for improving the computational performance is to exploit the inherent parallelism of GP by mapping the GP on parallel multiprocessor systems. Networks (clusters) of workstations (NOW) as a parallel environment feature most attractive specific cost characteristics

V.N. Alexandrov et al. (Eds.): ICCS 2001, LNCS 2073, pp. 284–293, 2001.
© Springer-Verlag Berlin Heidelberg 2001

due to the well-known fact that the prices of the components are based on the manufactured volume. Typically, NOW can be built of commodity "off-the-shelf" components, significantly reducing both the developing time and the costs [2]. Our *objective* is to develop MIGP featuring improved computational performance of GP by exploiting the parallelism among the evaluations of individuals. MIGP must be cost-effective in that to allow for deployment on clusters of commodity, non-dedicated workstations.

Although some work on implementing GP exploiting the parallelism among evaluations of individuals on NOW had been previously done [3], [4], [5], it assumes deployment of developed systems on networks of dedicated, homogeneous workstations. Therefore, the issues of resources location and allocation, scheduling, load balancing, and even more, the issue of creating a single image of the underlying distributed system had not been considered as relevant in these implementations. On the other hand, employing the currently available general-purpose full-featured metacomputer systems [6], [7], [8], [9] for addressing the concrete issue of parallel implementation of GP is viewed by us as too excessive approach. Moreover, although these systems do offer flexibility to the end-users, some concerns could be raised about their ability to be finely tuned to adequately address the communication-intensive nature of the considered model of parallelism in GP. In addition, from viewpoint of the communicating paradigm, the currently available metacomputers are heavily relying on the de-facto-standard MPI/PVM-based communications, while, in our view, the need for investigating the feasibility of recently emerged component object models for engagement as a communication paradigm in clusters seems to be underestimated by the HPC. All these concerns represent the rationale behind our approach to develop and evaluate the component object based SSIM for MIGP.

Addressing the challenge of potentially heterogeneous nature of resources in NOW, we propose a single system image middleware (SSIM), which represents the NOW as a single metacomputer (virtual supercomputer). Distributed component object model (DCOM), adopted as an underlying communicating paradigm contributes to address the heterogeneity of software platforms of workstations and the heterogeneity of network protocols. The resource management and scheduling subsystem of SSIM addresses the issue of efficient utilization of workstations that feature different speed of network connections, computational power, and/or dynamically changeable computational workloads. The reminder of the paper is organized as follows. Section 2 outlines the considered time series identification problem (TSIP), and the GP as an algorithmic paradigm, adopted for solving it. Section 3 discusses the developing of SSIM for MIGP. Also, it enumerates both the benefits and the challenges of adopting DCOM as a communicating paradigm. Section 4 presents performance evaluation results. The conclusion is drawn in Section 5.

2 The Problem and the Algorithmic Paradigm

Identification of a mathematical model of a real process is the necessary step in a wide class of modern control, information processing, diagnosis and related problem solving [10]. Let consider an identification problem statement. Assume that some plant is

isolated from the environment on the grounds of fixed features (Fig. 1). The input *u(n)* and output *y(n)* signals of the plant are supposed measurable at the instants n=1,2,… during functioning. The plant is interfered with random non-observable disturbance *(n)*. There is an unknown operator of input-output mapping of the form:

$$y(n) = A[u(n)] \quad (\mathbf{1})$$

The operator (1) is characterized with unknown structure *St* and parameter vector $\tilde{a}$:

$$A =< St, \tilde{a} > \quad (\mathbf{2})$$

The identification problem is determination of the plant's operator estimate $\hat{A}$ by means of input and output signals measuring and processing. Typically, the criteria to evaluate the quality of the model are based on error between plant's output and model's output. The identification is an optimization problem with an objective to find the minimum of the error between the output of the model and the real output. We consider the identification of the model of vibration data from the crystal-polishing machine as a real-world TSIP. Example of the sample data is shown in Fig. 2.

Requirements of high quality automatic control had made it necessary to use novel approaches for real process description which are often characterized with high dimensionality, substantial non-linearity and unmodelled dynamics. Such class of problems may be solved by soft computing approaches - artificial neural networks, fuzzy systems, chaos computing, immune network computing, genetic algorithms, GP, etc. As the results of many researchers have shown, effective systems may be designed on the basis of fusion of different soft computing approaches [11]. GP (Fig.3), considered in our approach, is an algorithmic paradigm, inspired by the natural evolution of species based on the survival of the fittest [1].

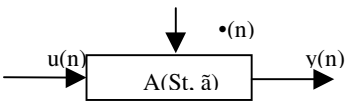


Fig. 1. Operator presentation of a plant

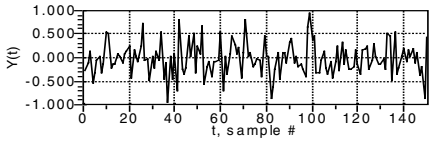


Fig. 2. Vibration data of crystal-polishing machine

```
1: Create the initial population of GPs;
2: Evaluate initial population;
3: while not success predicate do Steps 4..5
4:    Perform reproduction: mating pool creation;
5:    while initial population size not reached do Steps 6..9
6:       Select a pair of GPs from mating pool;
7:       Perform crossover and produce 2 GPs - offspring;
8:       Evaluate offspring;
9:       Incorporate offspring into the population;
```

Fig. 3. Algorithm of GP

GP considers a population of individuals (genetic programs, GPs) that evolves. Each of GPs represents a solution to the problem. The initially (randomly) created population evolves through many generations, applying the main genetic operations - reproduction, selection and crossover, at each generation until the best individual solves the problem with desired precision. The power of GP is in that computational effort of such evolution is much less than in the random search-based approaches. The main parameters of GP for TSIP are shown in Table 1.

3 SSIM for MIGP

3.1 Model of Parallel Implementation of GP

Partitioning of GP. Partitioning considers the way GP is decomposed into smaller, relatively autonomous processes that might be simultaneously executed on multiple workstations. Taking into consideration the algorithm of evolving single generation in GP and the runtime breakdown results obtained from the developed prototype of sequential GP for solving the TSIP, we considered the implementation of parallelism among the evaluations of individuals (Fig. 3, Step 8). The key argument in favor of such a decision is that for the considered case of TSIP the evaluation of individuals consumes more than 98% of GP-runtime.

Communicating Paradigm: DCOM. Exploiting the considered parallelism in GP assumes that the code of evaluation of individuals must be running on workstations, remote with respect to the workstation that manages the GP-population and performs the main genetic operations. Such a code must be implemented as a remotely controllable process, and several communication technologies can be exploited for such a purpose. These include component-object-based (COB) technologies, such as DCOM [12] and CORBA, and non-COB, such as sockets level programming, MPI, PVM, RPC, and Java RMI. Our choice for using COB paradigm is based on the fact that it provides the domain of distributed computing (DC) with many of the same benefits, such as encapsulation, reuse, portability, and extensibility, as it does for non-DC. Furthermore, applied to the domain of DC, these benefits can be extended with features as binary standardization, platform-, machine- and protocol-neutrality, and ability to seamlessly integrate with different Internet protocols. All these features are incorporated into DCOM, which in addition, as a system model offers generic support for the issues of naming, locating and protecting the communicating entities. Adopting DCOM tends to compromise (to certain degree) the communication characteristics of the implementation compared to the widely adopted MPI and PVM. The throughput of DCOM, which we obtained for two protocol stacks – TCP/IP (W2K) and UDP/IP (NT4.0), compared with the throughput of several implementations of MPI on NT4.0 [13] is shown in Fig. 4. The CPU for all of the considered cases is 450MHz Pentium II, and the underlying network is 100 Mbps Fast Ethernet. As figure illustrates, for small messages DCOM is more than 2 times slower than NT-MPICH – the fastest

implementation of MPI. For larger messages however, the throughput of DCOM on W2K (over TCP/IP) is only about 20% off the throughput of NT-MPICH.

Table 1. GP for TSIP

Objective	For given data source $Y(t_i)$, i=1,2,..2000 to find the analytical expression $Y(t_i)=F(S_T)$ which fits the given data source with specified error. S_T is the terminal set.
Terminal set	S_T =[$Y(t_{i-1})$, $Y(t_{i-2})$,…$Y(t_{i-P})$, C, A, M], where C is a random constant from the interval [0,1], M and A are the average and absolute average amplitude respectively. P=10.
Function set	[+, -, *, /]
Fitness cases	The given sample of 2000 data points.
Raw fitness	The average, taken over 2000 fitness cases, of quadratic value of the difference between value of the $Y(t_i)$, produced by individual (S-expression) and the target value $Y(t_i)$ of the given data source.
Reproduction ratio	20% - reproduction, 80% - selection (both – fitness-proportional)
Success predicate	Raw fitness is less than 0.01
Population size	1000 individuals

Challenges in Adopting DCOM. Batching. Implementing the proposed parallelism in GP exploiting DCOM implies that the evaluation of individuals is performed by remote DCOM-server, which runs on a separate workstation. The routine of evaluation of individuals is remotely invoked by the client (GP manager - GPM), which runs on separate client machine. Client submits the GPs to DCOM-server by invoking the corresponding methods of DCOM-server. The invocation is location transparent, platform-, and protocol-neutral, which allows for deployment of our implementation on network of heterogeneous workstations. However, the cost of these benefits is that, performance-wise compared to MPI and PVM, such a remote method invocation suffers from significant software overhead, and less data transmission rate (refer to Fig.4). For example, the typical software overhead of DCOM is in order of several hundreds •s, which is comparable with the computational cost of evaluating simple individuals.

We applied batching (i.e. bundling multiple method invocations into a single one) both in submission of the individuals and in forwarding the fitness values back to the client. The experimentally obtained effect of batching on the latency of submission of individuals with complexity of 100 tree nodes, which is close to the average for the GP-populations over many independent runs is shown in Fig. 5. As figure illustrates, with increasing of the batch size, the specific latency (*Ls*) of submitting single individual within the batch decreases, and for batch size of 40 individuals is about 8 times less than latency of submitting single individual *L(1GP)*. However, we consider the value of 16 individuals as an optimal batch size for our implementation. As depicted in Fig. 5, larger batches result only marginal improvement of the specific latency. In addition, larger batches imply increasing the granularity of scheduled tasks, which might increase the degree of unbalancing of the workload when the total amount of

batches cannot be evenly distributed among the available workstations. The effect of batching on the load balancing is considered later in following subsection 3.2.

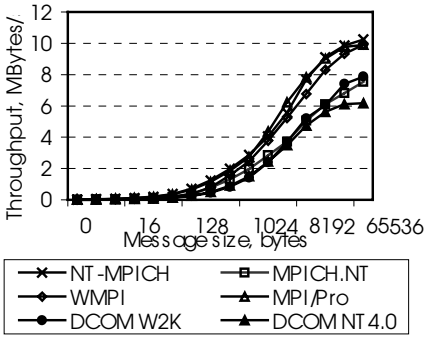


Fig. 4. Throughput of DCOM compared with various implementations of MPI on NT4.0

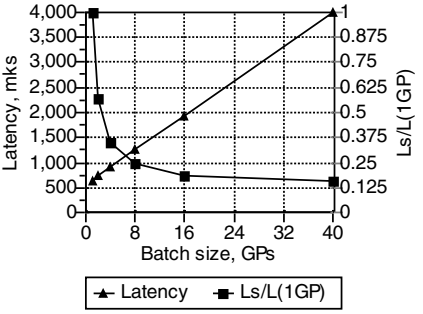


Fig. 5. Effect of batching on the latency of submitting GP-individuals

3.2 Distributed Architecture

Exploiting the considered model of parallelism implies that the routine of evaluation of the individuals is performed by Evaluation Servers (ES), running on separate workstations in the cluster. An eventual straightforward two-tiered implementation, where the client (GPM) besides the main genetic operations handles the resource management and scheduling (RMS) would raise modularity-, flexibility- and manageability related concerns. To address these concerns, we propose a three-tiered architecture incorporating the single system image middleware (SSIM) with functionality solely devoted to the issues of RMS. The architecture of MIGP is shown in Fig. 6. The functionality of the entities of the proposed architecture is as follows.

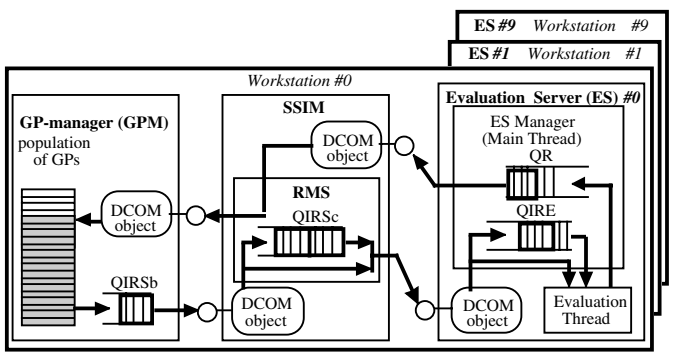


Fig. 6. Distributed architecture of MIGP

GPM. Manages the GP-population, and performs genetic operations – reproduction, selection, and crossover. Using a queue of individuals, ready to be submitted (QIRSb), packets the pairs of offspring into a single batch. Upon completing QIRSb submits the batch to SSIM. Incorporates the results of evaluation upon receiving them from SSIM.

SSIM. Dynamically locates and allocates the ESs. For GPM, SSIM creates a view of the pool of available ESs as a single server, respectively – the network of heterogeneous workstations – as a single metacomputer. Accepts the packets of offspring from GPM in the queue of individuals, ready to be scheduled (QIRSc) and assign each of them to specific ES in accordance with the scheduling policy. Accepts the results of evaluations and forwards them to GPM.

ES. Accepts the packets of offspring from SSIM in the queue of individuals ready to be evaluated (QIRE) and assigns the individuals to separate thread (evaluation thread - ET) for further evaluation. Introducing ET contributes to achieving asynchronous (non-blocking) SSIM – ES interaction by separating the communication from evaluation of the individuals. The queue of results (QR) accumulates the obtained fitness values prior to forwarding them in single batch back to the SSIM.

3.3 Functionality of SSIM

Resource Location and Allocation. The location of the ESs is performed assuming that ESs are installed on each of a priory known workstations in the cluster. The allocation of ES, performed by DCOM is initiated by SSIM during the creating an instance of the remote object. Not only to the issue of locating, creating instance of the object is also closely related to the issues of naming and protecting the communicating entities of the proposed distributed architecture. The benefits of adopting DCOM as a communication paradigm are that the latter, as a true system model uses globally unique identifiers to identify object classes and supported interfaces; encapsulates the life cycle of DCOM-server objects (ESs) via reference counting, and provides a means for secure access to objects and the data they encapsulate.

Scheduling and Load Balancing. The adopted scheduling policy is based on the combination of two strategies as follows. At the stage of evaluating of first batches of current generation SSIM applies static, round robin scheduling (RRS). At this stage SSIM feeds each ESs with 2 batches, providing ES-side overlapping of the evaluation with the communication. Then, in order to deal with eventually different, and dynamically changeable performance characteristics of ESs we propose a Synchronous Incremental Scheduling (SIS) – a dynamic scheduling policy, which allows for minimizing the computational and communicational cost of scheduling yet providing good quality of load balancing. Synchronously with the event of receiving the batch of solutions from ES, the scheduling subsystem of SSIM extracts the first batch from QIRSc and submits it to ES – the source of recently received result. However, even in homogeneous cluster, implementing adopted scheduling policy might not yield to perfect load balancing in case that the total amount of batches cannot be evenly distributed among the active ESs. Implementing batching would only deteriorate the load

balancing due to the increased granularity of the scheduled tasks. In order to estimate the optimal batch size we consider the upper bounds of the efficiency E of load balancing:

$$E = \frac{T_i}{T_a} = \frac{C \quad b/S}{C \quad ceil(b/S)} = \frac{b/S}{ceil(b/S)} \tag{3}$$

where T_i the GP-runtime for evolving single population for "ideal" load balancing when all the individuals are evenly distributed among ESs, T_a is the actual runtime for the best possible load balancing in homogeneous environment implementing batching, C is the computational cost of single batch, b is the amount of scheduled batches, S is the number of active ESs, and `ceil()` is a function that returns the smallest integer greater than or equal a specified value. The amount of batches b is:

$$b = P \quad R_{CO} / s \tag{4}$$

where P is the population size, R_{CO} is the crossover ratio, and s is the batch size. Applying (3) and (4) we obtain the values of E for the following values of the parameters: P=1000 individuals, and R_{CO}=0.8, S=var, and s=var. The values of E are shown in Fig.7. As figure illustrates, increased batch sizes yield lower values of efficiency. To maintain the bound of efficiency of load balancing equal to 1 we propose to dynamically adjust the population size in that the amount of batches, expressed in (4) could be evenly distributed among ESs. The correctness of this approach is based on the empirical observation that the computational effort of GP does not significantly change with small variations in population size. In our approach we consider the variation of population size of 6%, achieved for s=16 individuals as an acceptable in that it does not affect the computational effort of GP. Proposed method for dynamic adjustment of population size is implemented straightforwardly: SSIM includes the amount of currently active ESs in the batch of results, forwarded to GPM. The latter adjusts the population size in accordance with the following rule:

$$P^* = round \quad \frac{P \quad R_{CO}}{s \quad S} \quad \frac{s \quad S}{R_{CO}} \tag{5}$$

where s=16 individuals, R_{CO} and P are the end-user defined crossover ratio and population size respectively.

4. Performance Evaluation

The performance evaluation results are obtained from the developed prototype of MIGP running on local cluster of 10 W2K workstations. The workstations are 450MHz/64MB Pentium II-based Hitachi Flora. They are attached to 100 Base TX Ethernet through Intel 82558-based Ethernet adapter. Workstations are connected in network through Hitachi Summit-48 switching hub. Experimentally obtained speedup characteristics of MIGP are shown in Table 2. The granularity of parallelism derived from these data is about 100. The upper bound of the scalability (assuming that there are no network collisions, and no serialization of the ES-SSIM communications),

calculated as a ratio of the throughput of scheduling subsystem of SSIM to the performance of ES is more than 60, which indicates that one should expect close to linear speedup characteristics of MIGP deployed on the considered cluster of 10 workstations. The experimentally obtained speedup characteristics of MIGP, which are consistent with our anticipation, are shown in Fig.8. The speedup is derived as a ratio of runtime of evolving one generation in serial GP to the runtime of MIGP in configuration where each workstation (including workstation that hosts GPM and SSIM) hosts a single ES.

Table 2. Performance characteristics of MIGP

Characteristic	Value
Batch size, *individuals*	16
Computational cost of evolving one generation of individuals (for $P^*=1000$), *batches*	50
Runtime of evolving one generation in serial implementation of GP, *s*	14
Performance of serial implementation of GP, *batches/s*	3.6
Runtime of evolving one generation on MIGP configured with one remote ES, *s*	15
GPM: throughput of crossover operation, *batches/s*	2800
GPM: Runtime of submitting all the batches to SSIM, *s*	0.14
GPM-SSIM throughput, *batches/s*	370
Throughput of scheduling subsystem of SSIM (SSIM-ES throughput), *batches/s*	260
Performance of ES, *batches/s*	3.8

5. Conclusion

We presented a DCOM-based SSIM for MIGP, aimed to significantly improve the computational performance of GP exploiting the inherent parallelism of GP among the evaluations of individuals. MIGP runs on cost-effective clusters of commodity, non-dedicated, heterogeneous workstations. Developed SSIM represents these workstations as unified virtual resource and addresses the issues of locating and allocating the physical resources, communicating between the distributed entities, scheduling and load balancing. Adopting DCOM as a communicating paradigm offers the benefits of software platform- and network protocol neutrality of proposed implementation and generic support to the issues of naming, locating and allocating of the entities of MIGP. Batching reduces the specific software overhead of DCOM and increases the throughput of SSIM. Developed SIS features reduced runtime overhead and good quality of load balancing. Mechanism of dynamic adjustment of the size of GP-population is proposed as a solution to the problem of load unbalance when scheduled tasks cannot be evenly distributed among the available workstations. Presented results of experimentally obtained speedup characteristics show close to linear speedup of developed MIGP for solving TSIP on cluster of 10 W2K workstations.

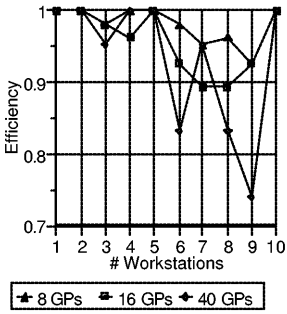


Fig 7. Estimated upper bound of the efficiency of load balancing for different system configurations. Obtained from (4)

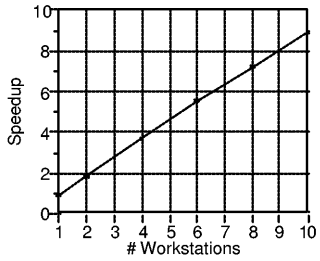


Fig. 8. Experimentally obtained speedup characteristics of MIGP for solving TSIP

References

1. Koza, J.R., Bennett III, F.H., Andre, D., Keane, M.A.: Genetic Programming III: Darwinian Invention and Problem Solving. Morgan Kaufmann Publishers, San Francisco, CA (1999).
2. Buyya, R.: High Performance Cluster Computing, Vol.1, Prentice Hall PTR, Upper Saddle River, New Jersey (1999)
3. Dracopoulos, D.C., Kent, S.: Speeding up genetic programming: A parallel BSP implementation. In: Koza, J.R., Goldberg, D., Fogel, D.B., Riolo, R.L. (eds): Genetic Programming 1996: Proceedings of the First Annual Conference, MIT Press, Stanford University, CA, 28-31 July (1996) 421
4. Oussaidene, M., Chopard, M., Pictet, O., Tomassini, M.: Parallel Genetic Programming and its Application to Trading Model Induction, Parallel Computing 23 (1997) 1183-1198.
5. Tanev, I.T., Uozumi, T., Ono, K.: DCOM-based Parallel Distributed Implementation of Genetic Programming, Parallel and Distributed Computing Practices Journal, Special Issue on Distributed Object Oriented Systems, 1, Vol.3 (2000)
6. Globus – http://www.globus.org/
7. Globe – http://www.cs.vu.nl/~steen/globe/
8. Legion – http://legion.virginia.edu/
9. Condor – http://www.cs.wisc.edu/condor/
10. Ljung L.: System Identification, Theory for the User. Prentice Hall, Englewood Cliffs, New Jersey (1987)
11. Akhmetov, D.F., Dote, Y.: Fuzzy System Identification With General Parameter Radial Basis Function Neural Network, In: Farinwata, S., Filev, D., Langari, R. (Eds.): Analytical Issues in Fuzzy Control: Synthesis and Analysis, John Wiley & Sons, Ltd, United Kingdom (2000) 73-92
12. Microsoft Corporation: COM Specification (1995), http://www.microsoft.com/com/ resources/specs.asp
13. Scholtyssik, K.: NT-MPICH – Project Description (2000) http://www.lfbs.rwth-aachen.de/~karsten/projects/nt-mpich/index.html

The Prioritized and Distributed Synchronization in Distributed Groups

Michel Trehel and Ahmed Housni

Université de Franche Comté, 16, route de Gray 25000 Besançon France,
{trehel, housni} @lifc.univ-fcomte.fr

Abstract. A simple and cheap algorithm is presented to allow prioritized mutual exclusion. There are several groups. All the members of a same group have the same level priority. Our algorithm is a token-based algorithm. Each group of participants (*site*) is represented by a tree structure. Inside a group, the requests are recorded in a global queue which circulates simultaneously and together with the token. The participant holding the token is the root of the tree linked to the router of the group. When a router transmits the token to another group, it preserves the role of the router in its group. The relation between routers is also represented by an rooted tree, the root is the router of the group holding the token. Besides a static logical structure similar to that used in Raymond's algorithm, our algorithm manages a global requester queue. If the requesting site and the owner of the token are in the same group, there is a reorganization of the group tree. If they are not in the same group, the tree of the requesting site and the tree of routers are reorganized. Algorithm in one group and extension to some groups are presented.

1 Introduction

Prioritized mutual exclusion can be applied to speech allocation in a multi-role conference. The speakers of each role constitute a group and there is a priority level by group. At a given moment, only one participant of only one group may speak to the other ones. When the situation is limited to two groups, prioritized mutual exclusion corresponds, for example, to a group of trainers and a group of trainees, in teleteaching. The resource may be the speech. Trainers have priority on trainees. Some works have been presented concerning the prioritized mutual exclusion. A. Goscinski's [1], algorithms lean on request broadcasts. The average number of messages is O(*n)* where n is the number of sites. K. Harathi and T. Johnson [2] propose prioritized spin lock mutual exclusion algorithms. The blocked processes spin on locally stored or cached variables. The performances are improved, compared with the previous ones. Nevertheless, it is O(n). T. Johnson and R. Newman-Wolfe [4] developed three algorithms giving approximately the same performances. They aimed at synchronizing the access to processor memory. One of them uses a rooted tree as in Raymond's approach [8].

V.N. Alexandrov et al. (Eds.): ICCS 2001, LNCS 2073, pp. 294-303, 2001.
© Springer-Verlag Berlin Heidelberg 2001

The other two algorithms use a path compression technique to reduce the number of messages. B. M. K. Qazzaz [5] gave an algorithm based on a binary tree. The prioritized nodes have a special position in the tree, what gives strong constraints to the system. Mueller [7] proposed an idea derived from the M. Naimi and M. Trehel [9] algorithm, associating a priority level with the requests. For that, the queue distributed in [9] is replaced in [7] by local queues. Every site owns a local queue. When it receives a request, it compares the priority of the queue's sites with the arriving request priority, to reorder the queue. The average number of messages is O(Log(n)). In our algorithm, there is a priority level by group. The presentation of the algorithm is modular: one group (all the same priority), and many groups (one priority level by group).

2 Our Algorithm for One Group

2.1 General Presentation

In this paper, the term "tree" is used when there is neither orientation neither root, and "rooted tree", when there is an orientation and a specific root. Both terms are included, because both concepts are needed. Our algorithm in one group is closed to K. Raymond's algorithm [8]. Let us describe the idea. The participants are structured as a logical tree. We will speak about the choice of the tree, function of performance considerations. A root is chosen in the initial situation. The root is said privileged because it owns the token. The choice of the root transforms the tree in a rooted tree. Every site owns a local variable called its "father", which indicates the direction of the root. "Father" is nil at the root. Look at the figure 1 as an example. Suppose the root A is in the critical section. If the nonprivileged node D wishes to enter the critical section, it sends a "request" message to its "father" C. When receiving the "request" message, the node C, if it is not the root, forwards the message to its "father" B. Thus a series of "request" messages travels along the path from the requesting node D to the root A. The message "request" is put in the memory of the root.

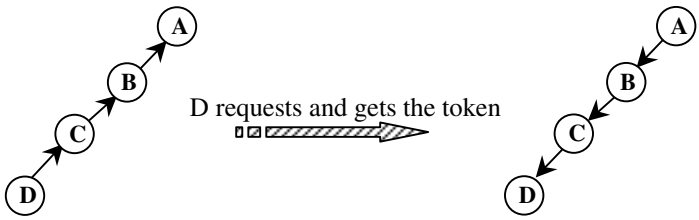


Fig. 1. Transformation

When the root releases the critical section, it sends the token to its neighbor B, in the direction of the requesting node. If A has more than a neighbor, a specific technique is necessary to determine to which node A must send the token. While sending the token, A looses its quality of root and its "father" becomes B. Now, the "father" of B is nil.

There is no change of the underlying tree, nevertheless the rooted tree is changed and the new root is B. If B is not the requesting node (B is not D), B forwards the token to its neighbor, in the direction of D. Thus a series of "token" messages travels from A to D. When it is arrived in D, D owns the token and enters the critical section. The rooted tree is reorganized and the new root is D.

2.2 Hypotheses

2.2.1. Network

The network is the initial rooted tree. The link between two sites is bi-directional, so that the communications can be sent after a reorganization of the rooted tree.

2.2.2 Communication

There is no loss, duplication or modification of messages. Transmission times are random, but finite. If several messages arrived simultaneously in a site, they are treated sequentially. Nevertheless a message can arrive when a site is in the critical section. From one site to another, the messages arrive in the order in which they have been sent. Communications run at least two times faster than the passage in critical section.

2.2.3 Critical Section

A site must wait reception of the critical section and releasing it before requesting it again. It stays in the critical section for a finite time.

2.2.4 Specifications

Mutual exclusion: Every site which requests the critical section has to obtain it after a finite time. However, it is not true when there are groups with different priorities.

2.3 Principles of the Algorithm

2.3.1 Token

It is a token-based algorithm. When the root releases the critical section, it sends the token in the direction of the requesting site. Every site crossed during the transmission of the token earns then loses the quality of the root.

2.3.2 Routing

The principle of the transmission of the request to the root was seen in the general presentation: the variable "father" gives the direction of the root. The token has to come back from the root to the requesting site. For that, the following technique is used: Every site i owns a routing table, i. e., it knows for every site j the first node on

the path from i to j in the tree. This principle is used in Internet (RIP). These tables remain unchanged. The following is an example of the routing table of A (table 1 and Figure 2).

Table 1. Routing table of A

A	*B*	*C*	*D*	*E*	*F*	*G*
nil	B	C	C	C	C	C

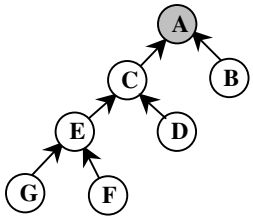


Fig. 2. the direction of the root

2.3.3 Queue

The requests are queued in the queue located at the root. When the root releases the critical section, it serves the head of the queue (FIFO service).

3 Example

E requests the critical section:

The variable "request" is true for E (Table 3). As E is not the root, it transmits the request to its father.

Table 2. Initial state of the system

	A	*B*	*C*	*D*	*E*	*F*	*G*
Father	nil	A	A	C	C	E	E
Request	F	F	F	F	F	F	F
Queue	nil	nil	nil	nil	nil	nil	nil

F: False, T: True, nil: empty

Table 3. E requests the critical section

	A	*B*	*C*	*D*	*E*	*F*	*G*
Father	nil	A	A	C	C	E	E
Request	F	F	F	F	T	F	F
Queue	nil	nil	nil	nil	nil	nil	nil

C receives the request from E and A receives the request from C:

As C is not the root, it transmits the request to its father. The table of variables is not changed. A puts E at the end of its local queue. The queue was empty, now it only contains the requesting-site. A is the root, is not in critical section (request = false), then its father becomes C which is the first node on the path to E (Table 4). The quality of rooted tree is lost for a short time. A is the father of C, C is the father of A). A sends the token and the queue to C to inform C that now it will be the new root.

Table 4. A has received the request of E

	A	*B*	*C*	*D*	*E*	*F*	*G*
Father	C	A	A	C	C	E	E
Request	F	F	F	F	T	F	F
Queue	E	nil	nil	nil	nil	nil	nil

Table 5. C has received the token from A

	A	*B*	*C*	*D*	*E*	*F*	*G*
Father	C	A	nil	C	C	E	E
Request	F	F	F	F	T	F	F
Queue	nil	nil	E	nil	nil	nil	nil

C receives the token and becomes the root:
C puts E in its local queue. The arrow "C to A" is now inverted. C is a temporary root (Table 5 and Figure 3). After this, the token is transmitted to E and the father of C becomes E. During a short time, the quality of rooted tree is lost (E is the father of C, C is the father of E). C sends the token and the queue to E.

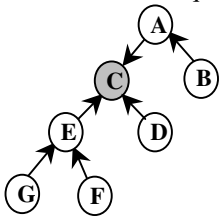


Fig. 3. C gets the token

E receives the token, becomes the root and enters its critical section:
The arrow "E to C" is now inverted. E becomes the root (Table 6 and Figure 4). C enters its critical section. If another site requests the critical section, the request will be transmitted to E.

Table 6. E receives the token

	A	*B*	*C*	*D*	*E*	*F*	*G*
Father	C	A	E	C	nil	E	E
Request	F	F	F	F	T	F	F
Queue	nil	nil	nil	nil	E	nil	nil

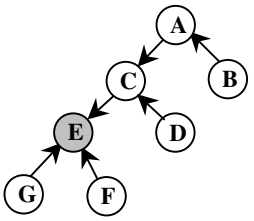


Fig. 4. E receives the token

4 Algorithm's Specification

The algorithm is presented as a series of procedures: "Requesting critical section", "Receiving request", "Receiving token", "Release critical section". A site can enter the critical section in "Requesting critical section" and "Receiving token". The procedure "Check the queue" details what to do after releasing the critical section. Precisely a site must check if another site has requested, and, in this case, it must send him the token.

```
Variables of site i

Const me =…{ identity of the site }
      N =… { total number of sites }
Type  Site = 1,…, N ∪ {nil} { nil means indefinite }
Var   Request: Boolean  {says if the site has requested}
      Father: site{Gives the organization of the rooted tree}
      Q-head, requesting-site: Site  {temporary variables}
```

```
      Routing: Array [Site] {Routing [j] is the first node on
                              the path from i to j}
      Queue: ordered set of [Site];
Type of messages   Req {transmission of a request}
                   Token {Transmission of the token}
Procedure Initialization
Begin
      Father := …{ nil for the root, else father }
      Request := false;
      Routing :… { the routing table is different for every
                   node}
      Queue :=  nil;
End
General procedures
      Put off (queue) { Put a site off the queue}
      Chain (queue, requesting site)
      { It is the  concatenation of a queue and a site}
Procedure Requesting critical-section
Begin
      Request := True
      If  (father = nil)  then
      begin
            Queue := {me}
            PERFORM CRITICAL SECTION
            RELEASE CRITICAL SECTION
      end
      Else
            Send (req, me) to father
      Endif
Endprocedure
Procedure Receiving request (req, requesting-site)
Begin
      If father = nil) then
            Begin
                  queue := Chain (queue, requesting-site)
                  If (not request) then {the root is not in
                        its critical section}
                  begin
                        father:= Routing (requesting-site)
                        Send (token, queue) to Routing (re-
                        questing-site)
                        Queue := nil
                  endif
            end
      else {The site has only to transmit the request}
            Send (req, requesting-site) to father
      endif
Endprocedure
```

```
Procedure Release critical-section
Begin
      Request := False
      Put off (queue)  {I have finished with the critical
                         section}
      CHECK THE QUEUE (queue)
Endprocedure

Procedure Check the queue (queue)
Begin
      If (not (queue = nil)) then
      Begin
            Q-head := head (queue)
            Father := Routing (Q-head )
            Send (token, queue) to father
            Queue := nil
Endprocedure

Procedure Receiving token (token, received-queue)
Begin
      Queue := received-queue
      father := nil
      If  me = head (queue) then
      Begin
            PERFORM CRITICAL SECTION
            RELEASE CRITICAL SECTION
      End
      else
            CHECK THE QUEUE (queue)
      endif
Endprocedure
```

5 Mutual Exclusion and Liveness Are Satisfied

Only a site owns the token and is authorized to send it to another one. That ensures mutual exclusion. The queue is FIFO. When a request is arrived in the queue, we are ensured that the corresponding site will obtain the critical section. It remains to check that every request arrives to the queue. That is not difficult because we are in a tree: there is no cycle. There is only a problem: a request can go from A to B during the time when token goes from B to A. That means the request does not take the shortest path, nevertheless it will arrive in the queue because the communications are two times faster than the passage in critical section (see hypotheses).

6 Study of Performances

Let us count the average number of messages in the case of one group for a particular rooted tree. The height of a node i is defined as the number of nodes from i to the root (i and the root included). This means the height of the root is 1. If the root requests the critical section, no messages are exchanged. When another node i, of height h_i, requests the critical section, the number of messages is 2 (h_i –1), i. e. (h_i –1) to transmit the request to the root, and (h_i –1) messages to transmit the token from the root to i. Suppose the probability of requesting the critical section, for node i, is p_i.

Lemma 1 : the average number of messages in a given rooted tree is: $2 \quad p_i (h_i \quad 1)$

Note If j is the root, (h_i –1) is equal to d (i, j) where d is the distance between the 2 nodes.

Corollary: the average number of messages in a given rooted tree is $2 \quad p_i d(j,i)$.

e.g., the average number of messages for Figure 5c is $2p_1+4p_3+4p_4+4p_5$.

It has been seen that, when the token has arrived at the requesting site, the tree is not changed, but there is another root. For instance, if 1 requests the critical section, the new rooted tree will be that one of Figure 5b. For a given tree, whatever the root, if i requests the critical section, the graph becomes a rooted tree of root i.

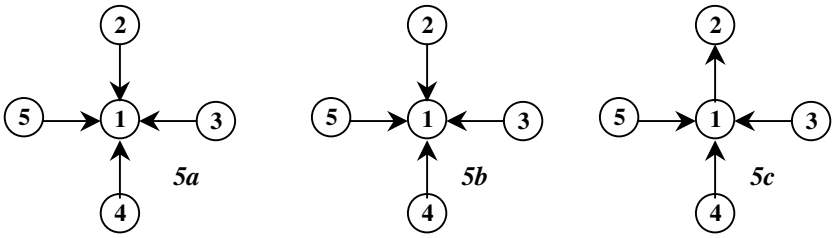


Fig. 5. Requests of 1 and 2.

This gives:

Lemma 2 For a given tree, the probability that the rooted tree of root i appears, is p_i. Then, this implies that the average number of messages of a tree will be the sum of the products of the probabilities of the rooted trees by the average numbers of messages:

Lemma 3 The cost (average number of messages) of a tree is: $2 \sum_{i=1}^{n} \sum_{j=1}^{n} p_i p_j d(i,j)$.

We have proved [3] that the cheapest tree is the star with the greatest probability at the center. The cost is less than 4 messages. However, star is not a good structure for fault tolerance, because the center is frequently impelled. Raymond obtains an average number of messages of O(Log (n)) by simulation, in the case of equiprobability.

7 Algorithm for n Groups

Maekawa [6] has written an algorithm by groups without priority. For us, groups are disjoint and every group has a priority level. When there are some requests in the queue, they are sorted in function of priorities. There is a router by group. The same structure (rooted tree) is defined between the routers than between the sites of a group. And the algorithm between the groups is nearly the same than between the routers of a group. The token is owned by the site at the root of the group, whose the router is at the root. When a site requests the critical section, it sends its request towards the root of its group. If the token is in the group, the request is put in the queue. A reorganization of the queue is processed in function of the site's request's priorities. When the site in critical section releases it, the token is sent to the first site of the queue (this site has the maximum priority). If the token is not in the group, the request is sent to the router of the group, which sends it to the group owner of the token (figure 6). The detailed program is given in [3].

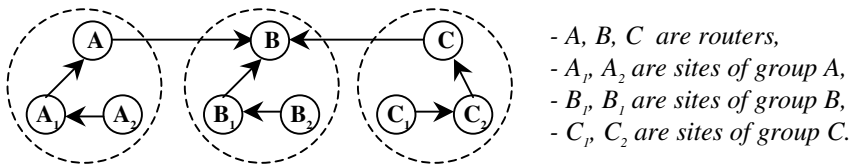


Fig. 6. Example for three structured groups.

8 Mutual Exclusion Is Satisfied

The proof of mutual exclusion is the same than in one group. Concerning deadlock, it is the same thing: deadlock is impossible. Nevertheless, there is absence of equity in prioritized mutual exclusion, because it is the aim of the system: it is possible that reordering the queue prevents a site to obtain the critical section.

9 Performances

There are thee kinds of messages : messages inside a group, messages from a router to another one, and messages between the routers and the sites. If the structure of group is a star, the cost is less than 4 messages. It will be the same for the communications between the routers. The result is that with a star for each group and a star between the routers, the number of messages is less than 12. That is really less than 12 if there are particular groups with great probabilities. Heterogeneity is a good factor of economy. Nevertheless, as for one group, the star is not a good structure for fault tolerance, because the center is frequently impelled.

10 Conclusion and Perspectives

The advantage of this algorithm is its simplicity, for one or many groups. A first objective is to validate the program. We think a proof with a validation tool will be simpler to obtain than with a mathematical proof. Concerning the performance, it is the same thing: performance by simulation will be simpler to obtain than mathematical performance. We have only considered the number of messages as measure of performance. We will have to consider also the efficiency for fault tolerance. That will give us completely different approach. There is always a problem concerning prioritized problems. It is possible that a site which requests critical section cannot obtain it. It would be interesting to give him the critical section after a maximum number of other sites. We intend to rewrite a new algorithm, to satisfy this specification.

References

1 A. Goscinski. "Two algorithms for mutual exclusion in real-time distributed computer systems", The Journal of Parallel and Distributed Computing, 9: pp.77-82, (1990).

2 K. Harathi and T. Johnson, "A priority synchronization algorithm for multiprocessors", Technical Report tr93.005. Available at ftp.cis.ufl.edu:cis/tech-reports, (1993).

3 A. Housni, M. Tréhel, "Specification of the prioritized algorithm for N groups", intern paper, Laboratore d'Informatique de l'université de Franche Comté, France, February (2001).

4 T. Johnson, R. E. Newman-Wolfe "A comparison of fast and low overhead distributed priority locks", Journal of Parallel and Distributed Computing 32 (1): pp.74-89 (1996).

5 B. M. K. Qazzaz "A new prioritized mutual exclusion algorithm for distributed systems", Doctoral Thesis, Dept of Comp. SCI., Southern Illinois University, Carbondale, (1994).

6 M. Maekawa, "A √N Algorithm for Mutual Exclusion in Decentralized Systems," ACM Transactions on Computer Systems, Vol3, pp. 145-159 (1985).

7 F. Mueller "Prioritized token-based mutual exclusion for distributed systems", 12th IPPS/SPDP, Orlando, Florida USA, (1998).

8 K. Raymond, "A tree-based algorithm for distributed mutual exclusion" ACM Transactions on Computer Systems Volume 7, Issue 1, pp.61-77 (1989).

9 M.Tréhel, M. Naimi, "A distributed algorithm for mutual exclusion based on data structures and fault-tolerance", 6th annual IEEE conference on computers and communications, Phoenix, Arizona, February (1987).

On Group Communication Systems: Insight, a Primer and a Snapshot

P. Gray†[1] and J. S. Pascoe‡

†Math & Computer Science
University of Norther Iowa
Cedar Falls, Iowa
50614-0506
gray@math-cs.cns.uni.edu

‡Department of Computer Science
University of Reading
Reading, UK
RG6 6AY
J.S.Pascoe@reading.ac.uk

Abstract. This paper contributes a concise introduction to the field of group communication systems and is structured as three integrated parts. In the first instance, this paper aims to share the practical insight gained from the implementation of several group communication projects back into the community. This is discussed in a form that can be used to guide and steer subsequent projects. Secondly, the paper aims to benefit newcomers to the subject by offering an introduction to some of the more pertinent areas of the field through a snapshot of its current state. The subjects of failure detectors, group membership (including virtual synchrony variants) and security are discussed. Although this paper presents a general view on these subjects, it alludes to the exemplars of the Collaborative Computing Frameworks (CCF) and IceT where necessary.

Keywords – group communication, state-of-the-art, tutorial, implementation, distributed computing.

1 Introduction

The field of group communications is a mature area of distributed computing. Projects such as the CCF [2], IceT [15], Totem [12, 3], InterGroup [7], Transis [4] and Horus [17] satisfy the design goals of many applications involving group communications. Numerous presentations of the theoretical work surrounding these projects has been contributed to the community. However, very little of the insight gained from the practical development of this work has been brought together and reintroduced to the field. Thus, this paper aims to present some of this practical insight in a form that can be used to guide future projects. In addition, this article offers a concise primer to the more noteworthy projects and texts that accompany the field.

The pertinent question of which topics to consider is now addressed. A comprehensive consideration of group communication issues would require a much fuller discussion than is given here. In this work, we aim to quickly introduce the reader to, what is in our opinion, the most general and fundamental concepts within the field. Thus, this paper is structured as follows. In section 2 we contrast the use of reliable multicast and keep-alive packets as failure detection mechanisms. Section 3 then discusses the implementation of group membership protocols, virtual synchrony variants and their associated repair algorithms. The issue of security in group communication infrastructures is considered in section 4 and finally, we give our conclusions in section 5.

[1] This research is supported by NSF grant ACI-9872167 and the University of Northern Iowa's Graduate College.

V.N. Alexandrov et al. (Eds.): ICCS 2001, LNCS 2073, pp. 307–315, 2001.
© Springer-Verlag Berlin Heidelberg 2001

1.1 Exemplars

In the interests of generality, the ensuing discussion remains at a high level. However, where necessary, we allude to the exemplar projects of the *Collaborative Computing Frameworks (CCF)* [2] and *IceT* [15].

The Collaborative Computing Frameworks The *Collaborative Computing Frameworks (CCF)*[2] is a suite of software systems, communications protocols, and methodologies that enable collaborative, computer based cooperative work. The CCF constructs a virtual work environment on multiple computer systems connected over the Internet, to form a *collaboratory*. In this setting, participants interact with each other, simultaneously access or operate computer applications, refer to global data repositories or archives and conduct a number of other activities via telepresence. CCF is an integrated framework for accomplishing most facets of collaborative work, discussion, or other group activity. It differs from other systems (audio tools, video/document conferencing) in that it aims to provide a comprehensive framework that addresses a broad range of collaborative issues. The CCF software systems are outcomes of ongoing experimental research in distributed computing and collaboration methodologies.

The *Collaborative Computing Transport Layer (CCTL)* [18] is the fabric upon which the entire system is built. A suite of reliable atomic communication protocols, CCTL supports *sessions* or heavyweight groups and *channels* (with relaxed virtual synchrony) that are able to exhibit multiple Qualities of Service (QoS) semantics. This in turn allows applications to operate across differing levels of QoS simultaneously. Unique features include a hierarchical group scheme, use of tunnel-augmented IP multicasting and a multithreaded implementation.

IceT The IceT project aims to provide an environment that is well suited for distributed computing as well as collaborative sessions. Groups of resources are *merged* together to form metacomputing and collaborative environments. One of the major attributes of the IceT environment is the ability for applications to be '*soft-installed*' on demand.

The commonality between distributed and collaborative computing issues in such an environment is the grouping of resources and the hierarchy generated by asymmetric access privileges. That is to say, if two researchers (collaborators) merge their resources together, their local resources typically extend certain privileges to their owners as opposed to remote users. Disk access, CPU usage, etc. are aspects of the environment that might be asymmetrically allocated.

In IceT, access and privileges are based on X509 certificates. Group definitions revolve around access certificates. A group of resources is identified by a common certificate. In the process of merging, where two groups of resources merge together to form a single virtual environment, individual certificates are used to generate common certificates[3] that represent the combined set of resources. The process of *splitting* a virtual environment involves heavy use of Certificate Revocation Lists (CRLs). When an environment is split apart, the certificate that identified the virtual composition is revoked, thereby revoking the privileges associated with the alliance.

[2] Further information is available at: `http://www.collaborative-computing.com/`.

[3] In actuality, a certificate *chain* is produced that consists of a new certificate and yet retains the qualities of the individual certificates.

2 Failure Detectors

A significant portion of recent work has addressed the impossibility of forming consensus on failed processes in asynchronous networks. This result stems from an inherent infeasibility to distinguish between failed processes and those that are arbitrarily slow. Chandra and Toueg [19] proposed eight classes of failure detectors defined in terms of *accuracy* and *completeness*. These are summarized in table 1 but for an interesting and more in-depth discussion, the reader is referred to [19].

The implementation of failure detection has often been based on the use of explicit keep-alive packets (or heartbeats). Indeed, Horus [17] and InterGroup [7] both use such a mechanism in their approaches to group communication. Heartbeat packets offer a number of advantages, the most notable being the detection of failed hosts within a known time window. However, a possible drawback of the mechanism is that they impose an unnecessary processing overhead in a failure free system. In wired environments, it is arguable that this overhead is insignificant, but in other networks (e.g. wireless) the processing requirement imposed by heartbeats can use significant levels of resources unnecessarily.

Another approach to failure detection is to augment reliable transmission mechanisms to act as weak failure detectors. Indeed host failures in CCTL are detected through the use of a reliable multicast primitive that has been augmented to provide a weak failure detector. This has the advantage that it does not incur any processing overhead on a failure free system, but a disadvantage of the mechanism is that a failed host can remain undetected for an arbitrary length of time.

The selection of an appropriate failure detector is dependent on the application. For high level collaborative environments such as the CCF, the timely detection of a failed host is often not imperative. Due to this, a failure detector based on a reliable transport mechanism may be adopted. However, in other group communication systems (e.g. safety-critical or process control systems), the detection of a node failure within a specific time period can be crucial to the provision of the service offered [16]. Thus, in this case, a mechanism based on heartbeats is usually more appropriate.

3 Group Membership

Group membership protocols are fundamental to any system consisting of a group of communicating processes. Due to network conditions, not all processes are able to communicate with others at all times. Processes can suffer crash failures or the network may partition. Thus, a process can not guarantee that the messages it sends are received by all members of the receiver set. This presents a new collection of failure scenarios that are challenging to deal with.

Completeness	Accuracy			
	Strong	Weak	Eventual Strong	Eventual Weak
Strong	*Perfect* $\mathcal{P}$	*Strong* $\mathcal{S}$	*EventuallyPerfect* $\diamond\mathcal{P}$	*EventuallyStrong* $\diamond\mathcal{S}$
Weak	$\mathcal{Q}$	*Weak* $\diamond\mathcal{Q}$	$\diamond\mathcal{Q}$	*EventuallyWeak* $\diamond\mathcal{W}$

Table 1. The Eight Classes of Failure Detector (see [19] for more details).

Membership protocols usually provide participant processes with the notion of a *view*, that is, an ordered list of the current members. All of the members in the current view are guaranteed to either accept the same view, or be removed from the session. Messages sent in the current view are delivered to the surviving members of the current view, and messages received in the current view are received by all surviving members in the current view. This is called *Virtual Synchrony* [9] and was first implemented in the Isis toolkit [8]. Since all members that can communicate see a failure at the same logical time, the resulting fault-tolerance protocols are significantly simplified.

3.1 Virtual Synchrony Variants

Virtual Synchrony is best understood as a simulation of fail-stop behavior, that is, members excluded from the session may still be alive. A number of Virtual Synchrony variants have been proposed and we consider two here.

Extended Virtual Synchrony The standard virtual synchrony model is inherently unequipped to deal with partitioning failures i.e. the model is defined in terms of a single system component in which process groups reside. Thus, in a *primary partition* system, only the fragment that resides in the primary component of the system can survive. The *Extended Virtual Synchrony* [12, 4] model suggests that applications can tolerate partitioning if processes are allowed to continue in non primary partitions. In such systems, any group of processes that can reach consensus on the membership of the partition is permitted to continue. However the remerging of such states is difficult and can not in general be automated.

Relaxed Virtual Synchrony In CCTL, the virtual synchrony algorithm was extended and optimized for a collaborative setting. To enforce virtual synchrony, all processes must receive the same messages in a given view. *Relaxed Virtual Synchrony* allows the sending and receiving views to differ. For example, in fig. 1 the receiving view of m contains d; the sending view does not. To guarantee virtual synchrony, the sender b compares the sending and receiving views for each message sent. Host b retransmits messages to processes in the receiving view but not in the sending view. If process b fails, all session members will eventually receive notification from the failure detector. Since b may have failed during a multicast operation, some channel members may not have received all messages sent by b. Before removing b, members must decide which messages from b are to be delivered. CCTL uses a flush procedure similar to that of Isis to perform this task. In brief, messages that have been received by all session members are said to be *stable*. Thus, the flush procedure removes all messages that are not stable. Virtual synchrony and extended virtual synchrony flush at each view change. CCTL only flushes in the event of a failure since relaxed virtual synchrony allows messages to be delivered in any view. Flushing is expensive as it delays view changes and delivery of subsequent messages. Since other systems generate significant numbers of view changes and also delay message transmission until after the flush terminates, relaxed virtual synchrony offers a somewhat more appropriate solution.

3.2 Membership Repair Algorithms

Membership Repair Algorithms (MRAs) are executed on the detection of one or more process crashes. As their name suggests, a membership repair algorithm will restore a group communication systems health in the presence of failures. This is often achieved by forming consensus on the failed members and then installing a corrected view.

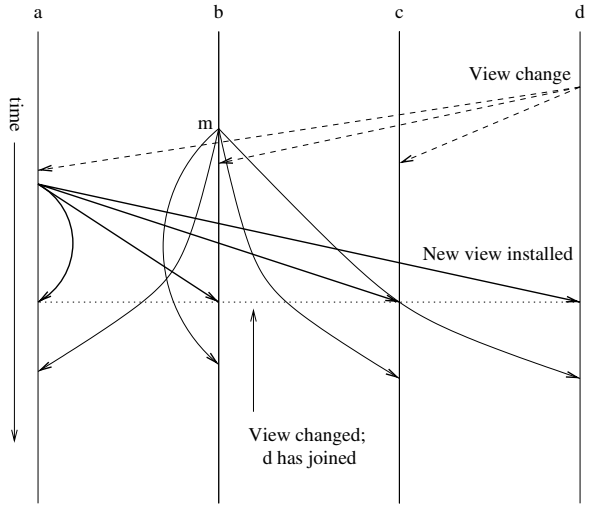


Fig. 1. Relaxed Virtual Synchrony: A view change is required as *d* makes a request to join the group. Thus, through relaxed Virtual Synchrony, there is no need to flush the system for the outstanding message *m* since the sending and receiving views are permitted to differ.

A number of novel membership repair algorithms have been suggested. In the CCF, membership repair is achieved through the use of an election based scheme. Each participant in a CCF session executes two threads. The *Error Monitor* is responsible for the logging of failures and the mediation of the more computationally expensive *Error Handler*. On detection of a threshold number of consecutive failures, the Error Handler is triggered and a message is multicast to all of the sessions participants informing them that an election is about to take place. Each participant then probes the network to ascertain an up-to-date snapshot of the sessions liveness. These views form the basis of votes which are relayed to the most senior session member (or *Error Master*) who collates and sorts the votes into a decision on which participants should be removed from the session. This result is multicast to the remaining live session members and the new view is installed.

The Totem Membership Repair Algorithm The Totem MRA is in some respects similar to the strategy employed in the CCF viz. an indication of failure is denoted by a predetermined number of outstanding messages. In Totem, when a failure is detected by a processor, the identification number of the suspect process is added to that processors *fail* set. The processor then enters the *Gather* state in which consensus is formed and the corrected view is installed. The algorithm is optimized to yield as large a membership set as possible, whilst not invalidating its *termination* requirement [1]. For a discussion on additional issues such as token loss and valid membership changes during the Gather state, see [3].

The InterGroup Membership Repair Algorithm InterGroup [7] implements an interesting MRA that is optimized for scalability. The algorithm is divided into two protocols: a lightweight version for passive receivers and a more expensive component that is executed by those processes that are members of the sender set, or wish to become

members of the sender set in the next view. This approach relies on the assumption that the majority of processors are passive, which given the intended applications of InterGroup, is often the case. Thus, the InterGroup MRA is, in its selected context, an effective method..

4 Security

In the context of group-based communication systems that are built upon insecure communication substrates, it is often essential to have security mechanisms in place. This includes addressing aspects of data encryption, mechanisms to authenticate processes and memberships in designated groups, and a way to digitally sign messages.

An associated aspect of security in the context of group-based communication systems is the issue of scalability. The protocol used for scalability should not become an obstacle for 'modest' group memberships. This is especially true in the context of cluster computing and groupware applications, where both the number of interprocess communication channels and the latency of the network are significant security factors.

The topic of security is quite an active research area in general. For a background on general principles and algorithms, see [10], [20], and [5].

4.1 Key-Pair and Symmetric Ciphers

One of the fundamental notions to the aspect of security aims at securing private information over a public channel. Briefly, there are two well-known and widely used approaches to this end; *key-pair* and *symmetric* ciphers. A *cipher* is an algorithm that takes the raw form of the information, or *plaintext*, and transforms it into *ciphertext*, or information that is unintelligible to an outside party.

A *key-pair* cipher is an algorithm that is based upon the use of two data sets, or *keys*. Both keys are required for the process of enciphering and deciphering plaintext. The security of the algorithm hinges upon maintaining close control of one of the keys. This key is known as the *private* key, and is never disclosed to any outside party. The other key, known as the *public* key, is freely distributable and open for public inspection. Since both keys are required for encrypting and decrypting information, possession of the public key will not be sufficient enough to compromise the communication process. Given a communication channel between Mary and Bob, Mary can convert plaintext into ciphertext using Bob's public key. This ciphertext would only be able to be deciphered by someone possessing Bob's private key. If Bob's private key is known only to Bob, then Bob would be the only party able to decipher the message. Similarly, Bob could respond to Mary in a secure manner by enciphering plaintext with Mary's public key, etc.

One of the drawbacks of the key-pair approach to encryption is the amount of overhead incurred in the

$$\text{plaintext} \xrightarrow{\text{key 1}} \text{ciphertext} \xrightarrow{\text{key 2}} \text{plaintext}$$

conversion. A more efficient algorithm is based upon the use of a single *session* key for both the encipher and decipher processes. That is:

$$\text{plaintext} \xrightarrow{\text{symmetric key}} \text{ciphertext} \xrightarrow{\text{symmetric key}} \text{plaintext}.$$

The generation of session keys is often facilitated by an initial key-pair based communication. Examples of these ciphering approaches are given below.

4.2 Certificate Based Security

Certificate-based systems rely on Certificate Authorities (CA) and trusted intermediaries to establish authentication of both clients and servers. In a certificate-based system, key-pairs are used to encrypt and decipher messages, to digitally sign message content, and for authentication. Key holders, clients and servers for example, are held responsible for keeping their *secret* key private. That is, from being compromised. The other key, the *public* key, is freely open to public inspection. These keys work together for encrypting and decrypting messages. Which key is used for encryption or decryption depends upon the final recipient of the message. Typically, a message originating from a key holder would be encrypted with the private key. The recipient would decrypt the contents with the key holders public key complement.

In a certificate based system, a Certificate Authority (CA) is presented with a key holder's public key and would issue a relatively long-lived credential known as a public key certificate. When two key holder's possess such certificates, they can authenticate each other without further reference to a CA. However, due to the long-lived credential duration, the CA's Certificate Revocation Lists (CRL's) play a role to update revoked credentials. In terms of scalability, a certificate-based system would be limited by the frequency with which group members would need to contact the CA's CRL to learn of revoked certificates.

4.3 Key Distribution Centers

Assume that security is based upon a single, *secret* key technology. That is, where two parties hold the same key for use in encryption and decryption of messages. If the group membership grows fairly large, say n then each group member would need to know $n-1$ keys. Further, the addition of a group member would constitute the generation of n keys. The keys would also have to be able to be distributed securely amongst the group members, which soon becomes an overwhelming task except for all but small group memberships.

One way to make things more manageable is to use a single trusted node known to maintain the keys for all the nodes. This node is called the *Key Distribution Center*, (KDC). Individual channels are secured between group members by first contacting the KDC for a *ticket-granting ticket* (TGT), which must then be presented to a centralized Ticket Granting Service (TGS) to obtain a session ticket for the connection. Relative to certificates, the TGT's are short lived, and provides the mechanism to guard against revoked credentials.

Kerberos [13] is an example of an authentication scheme implementation that is built upon the KDC paradigm, supporting many network applications and services. The KDC's make key distribution much more convenient, but there are some major disadvantages too. Mainly, the KDC has enough information to masquerade as any of the

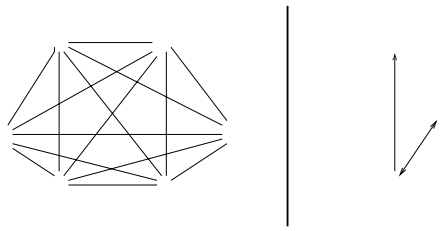


Fig. 2. Using unique session keys for each possible communication channel (left) scales poorly to large groups. Instead, the Ticket Granting Service is used to maintain and distribute session keys on demand (right).

groups members. That is, if it were to be compromised, every connection would be vulnerable. It is a single point of failure in the scenario as well, since if the KDC were to become unreachable, subsequent communication within the group structure would not be possible.

These issues, have to a large extent, been addressed in work such as the Ensemble project. Ensemble has incorporated public key and KDC features into a robust security framework that allows for multiple group partitions, group re-key on demand, and supporting group libraries for building applications that utilize the Ensemble authentication framework. For more information on Ensemble, consult [14].

5 Conclusion

This paper has presented tutorial style insight into the central issues surrounding group communication systems. The above discussion is presented in a form that is suitable for new researchers to the field. Where possible, the discussion has attempted to reintroduce some of the implementation insight gained from the development of the wealth of software in the area. In the future, it is envisaged that group communication systems will benefit from the recent advances in the fields of wireless networking and mobile computing. Thus, the consideration of the fields fundamental ideas in these novel contexts poses a series of new and interesting challenges to both the established member of the field, and the newcomer.

5.1 Recommended Reading

We briefly draw attention to some of the more noteworthy references in the field. In the authors opinion, the most comprehensive reference on group communication systems is Birman [9]. More detail on Isis, Virtual Synchrony and Hours can be found in numerous papers, available from Cornell's web site[4]. On specifically the concept of Extended Virtual Synchrony, the reader should consult Agarwal's thesis [3]. For security issues, consult the Ensemble project [14]. For *synchronous* group communication systems, the interested reader is encouraged to study the work of Kopetz [11]. Researchers interested in studying failure detectors should begin by reading the seminal Chandra and Toueg paper [19]. Finally, the work of Holzmann on the SPIN protocol validator [6] provides an interesting starting point for any researchers who are interested in the more formal aspects of the field.

[4] See: `http://www.cs.cornell.edu/Info/Projects/HORUS/`.

References

1. H. Attiya and J. Welch. *Distributed Computing: Fundamentals, Simulations and Advanced Topics*. McGraw-Hill, 1998.
2. S. Chodrow, S. Cheung, P. Hutto, A. Krantz, P. Gray, T. Goddard, I. Rhee, and V. Sunderam. CCF: A Collaborative Computing Frameworks. In *IEEE Internet Computing*, January / February 2000.
3. D. A. Agarwal. *Totem: A Reliable Ordered Delivery Protocol for Interconnected Local-Area Networks*. PhD thesis, University of California, Santa Barbara, August 1994.
4. D. Dolev and D. Malki. The Transis Approach to High Availability Cluster Communication. In *Communications of the ACM*, April 1996.
5. Whitfield Diffie and Martin Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, IT(22):644–654, November 1976.
6. G. J. Holzmann. *Design and Validation of Computer Protocols*. Prentice Hall, 1991.
7. K. Berket. *The InterGroup Protocols: Scalable Group Communication for the Internet*. PhD thesis, University of California, Santa Barbara, December 2000.
8. K. P. Birman. The Process Group Approach to Reliable Distributed Computing. *Communications of The ACM*, pages 37–53, December 1993.
9. K. P. Birman. *Building Secure and Reliable Network Applications*. Prentice Hall, 1997.
10. Charlie Kaufman, Radia Perlman, and Mike Speciner. *Network Security: Private communication in a public world*. Prentice Hall, Upper Saddle River, New Jersey 07458, 1995.
11. H. Kopetz. *Real-Time Systems: Design Principles for Distributed Embedded Applications*. Kluwer Academic Publishers, 1997.
12. L. E. Moser and P. M. Melliar-Smith and D. A. Agarwal and R. K. Budhia and C. A. Lingley-Papadopoulos. Totem: A Fault-Tolerant Multicast Group Communication System. In *Communications of the ACM*, April 1996.
13. B. Clifford Neuman and Theodore Y. T'so. Kerberos: An authentication service for computer networks. *IEE Communications*, pages 33–38, September 94.
14. O. Rodeh and K. P. Birman and D. Dolev. The Architecture and Performance of Security Protocols in the Ensemble Group Communication System. Technical Report TR2000-1791, Cornell University, March 2000.
15. P. Gray and V. S. Sunderam. IceT: Distributed Computing and Java. *Journal of Concurrency: Practice and Experience*, 9(11):1161–1168, 1997.
16. J. S. Pascoe and R. Loader. A Survey on Safety-Critical Multicast Networking. In *Proc. Safecomp 2000*, October 2000.
17. R. van Renesse and K. P. Birman and S. Maffeis. Horus, A Flexible Group Communication System. In *Communications of the ACM*, April 1996.
18. I. Rhee, S. Cheung, P. Hutto, A. Krantz, and V. Sunderam. Group Communication Support for Distributed Collaboration Systems. In *Proc. Cluster Computing: Networks, Software Tools and Applications*, December 1998.
19. T. D. Chandra and S. Toueg. Unreliable Failure Detectors for Reliable Distributed Systems. *Journal of the Association for Computing Machinery*, 43(2), 1996.
20. Mary Thompson, William Johnston, Srilekha Mudambai, Gary Hoo, Keith Jackson, and Abdelilah Essiari. Certificate-based access control for widely distributed resources. In *Proceedings of the Eighth Usenix Security Symposium*, August 99.

Overview of the InterGroup Protocols*

K. Berket[1], D.A. Agarwal[1], P.M. Melliar-Smith[2], and L.E. Moser[2]

[1] Ernest Orlando Lawrence Berkeley National Laboratory
1 Cyclotron Rd, MS 50B-2239
Berkeley, CA 94720
{KBerket, DAAgarwal}@lbl.gov
[2] Department of Electrical and Computer Engineering
University of California, Santa Barbara
Santa Barbara, CA 93106
{pmms, moser}@ece.ucsb.edu

Abstract. Existing reliable ordered group communication protocols have been developed for local-area networks and do not, in general, scale well to large numbers of nodes and wide-area networks. The InterGroup suite of protocols is a scalable group communication system that introduces a novel approach to handling group membership, and supports a receiver-oriented selection of service. The protocols are intended for a wide-area network, with a large number of nodes, that has highly variable delays and a high message loss rate, such as the Internet. The levels of the message delivery service range from unreliable unordered to reliable group timestamp ordered.

1 Introduction

Distributed applications often need to maintain consistency of replicated information and coordinate the activities of many processes. Collaborative applications and distributed computations are both examples of these types of applications. With the advent of grids [8], distributed computations will be spread across multiple computer centers requiring efficient mechanisms for coordination between the processes. Collaborations are by their very nature distributed and built in an incremental, ad hoc manner. Group communication provides a very natural mechanism for supporting these types of applications and allowing them to use a peer-to-peer architecture rather than a server-based architecture.

The MBone videoconferencing tools (vic, vat, and rat), the session directory tool (sdr) and the whiteboard tool (wb)[1] are excellent examples of the peer-to-peer model. These tools are designed to use multicast protocols to send data, which allows groups to form and communicate without coordination with

* This work was supported by the Director, Office of Science. Office of Advanced Scientific Computing Research. Mathematical, Information, and Computational Sciences Division, U.S. Department of Energy under Contract No. DE-AC03-76SF00098.

[1] More information on all of these tools and their binaries can be found at http://www-itg.lbl.gov/mbone.

V.N. Alexandrov et al. (Eds.): ICCS 2001, LNCS 2073, pp. 316–325, 2001.
© Springer-Verlag Berlin Heidelberg 2001

a server. This peer-to-peer model inherently makes the tools easier to design and to operate for groups of two and groups of hundreds. Because there are no servers, groups can be formed in an ad hoc manner with no setup. Scheduling using a centralized authority is sometimes used in these systems.

There are many applications that can benefit from the use of a peer-to-peer group communication capability. Instant messaging systems, shared remote visualization, shared virtual reality and collaborative remote control of instruments are just a few examples. Most of these applications currently use a central server to collect messages and forward them to the participants. A peer-to-peer group communication service that provides group membership services, reliable ordered message delivery, and progress in the presence of process and network faults can be used to allow the participants to talk directly to each other.

Although group communication systems can provide these services, the protocols have historically been limited in their scalability. The InterGroup protocol suite, described in this paper, is a group communication system that tackles scalability by taking a novel approach to providing these services. Our solution includes redefining the meaning of group membership, allowing voluntary membership changes, adding a receiver-oriented selection of delivery services (which permits heterogeneity of the receiver set), and providing a scalable reliability service. The InterGroup protocols are designed specifically with the intention of scaling to the Internet and to large numbers of participants.

2 Related Work

Group communication systems provide reliable group ordered message delivery and membership services that allow the system to make progress in the presence of process and network faults. The main concerns in scaling these protocols are flow control, congestion avoidance, the reliable multicast service, and the membership service. The membership service needs to support a reliable group-ordered delivery service and a form of virtual synchrony [5,12]. Because of this, it is essential for the membership service to be based on protocols that reach group-wide consistent decisions.

Traditionally, group communication systems, such as the Totem Single Ring Protocol (SRP) [3], Transis [2], Isis [5], and RMP [18], were designed for the local-area network environment, where network latencies and losses are minimal, and there is a small number of processes. Because of this, scalability concerns were not vital in the design of these systems.

Recently, group communication systems have made advances beyond the single local-area network environment, and research into scaling the membership services has intensified. The Totem Multiple Ring Protocol (MRP) [1] uses a hierarchy of rings interconnected by gateways. Each ring is an instantiation of the Totem SRP and the gateways provide coordination and message forwarding between the rings.

To reduce the costs encountered in applications requiring the use of a large number of process groups, dynamic light-weight groups [9] were introduced. The

idea is to map this large number of application process groups to a smaller number of protocol process groups. This concept is implemented by the process group interface of Totem [13], which provides a static mapping of the application groups to one protocol group, and by the gateways of the Totem MRP, which filters the forwarding of messages based on application groups.

Another trend in group communication systems has been the breaking up of the system into building blocks. Horus [17] (and its follow-on Ensemble[16]) introduced the building block approach to group communication systems. This approach allows more flexibility in the delivery services provided to the application. It also breaks away from a monolithic approach to design, allowing a better understanding of the interactions within a group communication system.

Moshe[10] is a group membership service (building block) for use by group communication systems in the wide-area environment. It provides an optimistic algorithm, for reaching a group-wide consistent decision regarding the membership, that usually finishes in one round. This is achieved by separating the membership service from the fault detection mechanisms in such a way that most membership changes can be handled as voluntary. Moshe also separates the membership service from the virtual synchrony service. For full virtual synchrony, an additional layer built on top of Moshe is necessary.

Scalable Reliable Multicast (SRM) [7] is a protocol that was designed specifically to scale to the Internet. It is not a full group communication system; it only provides the mechanisms for recovering messages in a scalable manner. It achieves this by separating the reliability mechanisms from the loss detection mechanisms, leaving those up to a higher layer. The SRM protocol exchanges *session messages* between group members to update the control information at each individual process. The original SRM protocol uses a group-wide multicast for all of its communication, which limits its scalability. One proposed solution to this problem is the organization of the group members in a self-configuring hierarchy [15].

3 The Architecture

The InterGroup protocols are designed using a building block approach. We divided the protocols into four separate modules based on functionality. The modules are control hierarchy, reliable multicast, message delivery, and process group membership.

The control hierarchy provides a scalable mechanism for the exchange of control information between sites in the system. Each site has a control process that collects and disseminates the control information for all of the processes at that site. The control hierarchy also provides mechanisms for determining message stability, and providing group-wide consistency of information to the group members. The essential components of the control hierarchy are explained in Section 4.

Reliable multicast provides mechanisms to detect missing messages, request the retransmission of messages, retransmit messages, and detect whether a mes-

sage can be recovered. The detection of missing messages in InterGroup is through detection of gaps in the sequence numbers of messages from the same source. When a node detects a missing message, it requests the retransmission of the message.

Message delivery entails the ordering and delivery of messages to the application based on the delivery service chosen by the application for a process group. The essential components of message ordering and delivery are explained in Section 5.

The process group membership protocols run at each process and track the changes in the group membership. They are affected by the delivery service chosen by the application and the sending characteristics of the process. The essential components of process group membership are explained in Section 6.

4 Control Information

There are many types of control information that need to be gathered by the InterGroup protocols. The reliable multicast protocols gather information about the latency between processes (based on algorithms introduced in SRM). The buffer management protocols gather information from the reliability service of each process so messages are held in the buffers only until they have been delivered to all of the processes in the group. The membership protocols collect information from all of the processes in the group in order to reach a consistent decision regarding the group-wide logical time at which a membership change occurs, called a *cut*.[2]

This control information must be obtained from all of the processes in the group or from all of the processes in the system. To make this operation more scalable, we gather and disseminate the control information in a hierarchical manner. The structure of our hierarchy is based on the work proposed for scaling the control information exchange in SRM [15]. The logical structure of this hierarchy attempts to mimic the underlying network topology by considering the latencies between control processes. This structure improves the efficiency of the control communication.

The control processes are organized in multicast trees, with the roots referred to as *coordinators*, and the leaves referred to as *children* (Fig. 1). Each child is associated with at least one coordinator. The *local group* of a coordinator is composed of the children of that coordinator (including the coordinator itself). The *coordinator group* consists of all the coordinators. Each control process limits its communication of control information to its local group; the coordinators also communicate with the coordinator group. The frequency of control messages is regulated using a simplified version of the congestion control algorithm used by RTCP [14].

The hierarchy dynamically reorganizes based on changes in the system. A self-determination protocol is executed periodically to determine whether a control

[2] This is required for virtual synchrony.

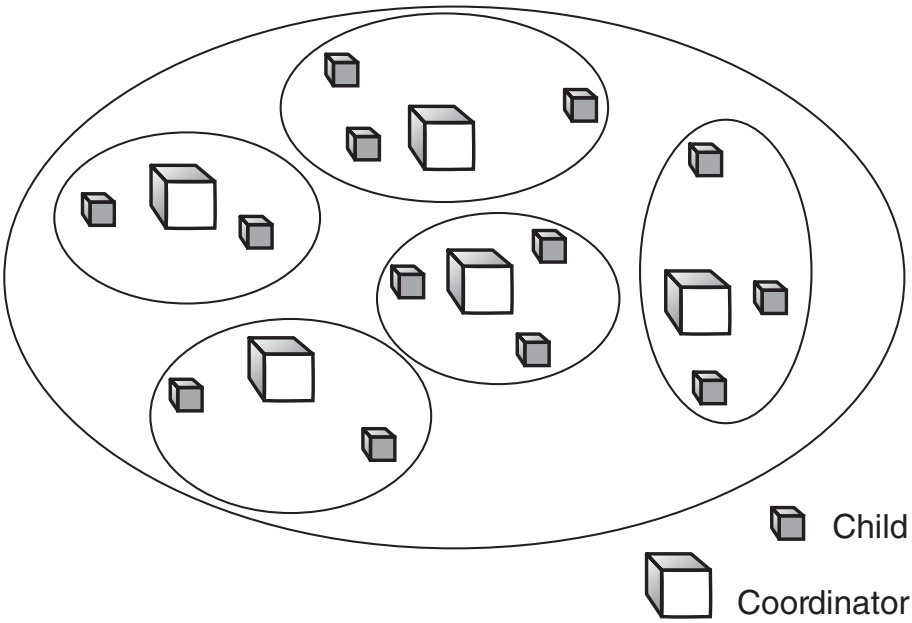


Fig. 1. The control hierarchy.

process in the hierarchy should change states (child to coordinator or vice versa) in response to changes in the system. The determination of a state change is a local decision made at each control process based on control information gathered before the self-determination protocol is executed, and based on a predefined set of rules adapted from [15].

A control process, upon startup, checks to see how many coordinators are present in the hierarchy, by checking the messages sent in the coordinator group.[3] If the number of coordinators is less than or equal to the expected average coordinator group size or the control process does not receive a control message from the coordinator group within a given time, the control process starts up as a coordinator. Otherwise, the control process starts up as a child.

A control process, starting up as a coordinator immediately starts sending control messages. A control process, starting up as a child, needs to find a coordinator that will accept it into its local group. This step is accomplished by using an expanding ring search. When it finds a coordinator, the control process starts sending messages to the coordinator's local group and becomes a child. If the expanding ring search doesn't provide a coordinator within a given time, the control process starts up as a coordinator.

The detection of control process faults is accomplished via a fault-detection algorithm that runs periodically. If the information from a control process has not been updated recently, the control process is removed from the data structures

[3] Only one message is necessary to make this determination.

and thus removed from this control process's view of the membership of the hierarchy. When a fault regarding the coordinator for this control process is detected, the control process uses the information from the self-determination protocol to decide whether it should find another coordinator and remain a child, or whether it should become a coordinator.

Global control traffic from all control processes is gathered using the hierarchy. The control information is aggregated as it is gathered through the InterGroup control hierarchy, thus controlling the control information message size. Otherwise, the size of control messages would grow proportionally with the number of control processes.

For a detailed description of the control hierarchy and a full state machine, see [4].

5 Delivery Services

The InterGroup system provides the following delivery services within a process group:

1. **Unreliable unordered**. Messages received from the process group are delivered directly to the application. Some messages might never be received, and multiple copies of the same message might be received. There is no guarantee regarding the order in which messages are received. IP Multicast provides this functionality.
2. **Reliable source ordered**. All messages from a particular source will be received by the application (unless a process fault occurs), and they will be delivered in sequence number order. This service is well-suited to applications such as multicast file transfers and many other applications currently using TCP/IP [6].
3. **Reliable group timestamp ordered**. Messages are received by the application in timestamp order over the entire process group. The membership service ensures the consistency of received messages at group members during membership changes. This service is closest to the idea of agreed messages in group communication systems [3].

Each process determines the delivery service it desires for a process group at the receiving end. To achieve this receiver-oriented delivery service, we need (1) to order messages independently at each receiver without restricting the delivery service choices of the other receivers, and (2) to separate the reliability service from the ordering service.

We make application messages "born-ordered" [12] to achieve the first goal. Each message is ordered based on information in the message: the process identifier of the sender, a sequence number, and a logical timestamp at the time the message is sent. The process identifier is guaranteed to be unique in this group. The sequence numbers preserve the order of the messages sent by the application. The logical timestamp is based on a Lamport clock [11], and is used to preserve the causality between messages in the group. These three values are

used by a deterministic algorithm applied at the individual receivers to produce a group-wide ordering of messages. Determining message order during a membership change is the work of the protocols that guarantee virtual synchrony (see Section 6).

The reliability service in the InterGroup protocols is represented by the reliable multicast module. The ordering protocols receive messages in sequence number order for each individual source from the reliability service. In the case that an unreliable delivery service is requested, the reliability service is bypassed.

These mechanisms allow the delivery service provided to the application to be chosen, independently, at each receiver. Our approach requires all of the active senders in the group to subscribe to the reliable group timestamp ordered delivery service[4] to preserve causality between messages. This requirement results in unnecessary overhead if none of the processes receiving messages in the process group has subscribed to the reliable group timestamp ordered delivery service. However, the benefits are the flexibility and data abstraction that this method provides.

A benefit of the receiver-oriented selection of delivery service is that the number of participants in the acknowledgment and cut gathering operations can be reduced. Processes that have not subscribed to the reliable group timestamp ordered delivery service do not, in general, have to participate in these operations.

6 Membership

The process group membership is used to update the group membership and allow the delivery of messages to continue after faults, joins and merges.

The InterGroup group communication system takes a novel approach to providing consistent message ordering and delivery within groups. A cornerstone of this approach is the recognition that the message order and reliability constraints can be met by counting only the processes currently sending messages in the group. In the InterGroup system, not all processes are equal. In each process group, a process is classified by its recent activity. If the process has been sending data to the group recently, it is classified as an *active sender*. Each group thus has two memberships; the *receiver membership* that contains all the members of the group, and the *sender membership* that contains only the active senders. The sender group membership is maintained using consistency-based membership mechanisms and is explicitly known. The receiver group membership does not need to be maintained explicitly for the purposes of message ordering and reliable delivery.

The active senders run a membership repair algorithm (MRA) that is based on the membership algorithms used by Transis [2]. The MRA of the InterGroup protocols has been designed so that participation of processes not in the sender group is minimized. The active senders run the membership algorithms to reach a

[4] This does not affect the application requested delivery service. It is handled internal to the InterGroup protocols.

consistent decision on the new membership, to ensure that a unique cut is chosen, and to decide on the place in the message flow that a membership change occurs.

Of the remaining processes, only the processes that have requested the reliable group timestamp ordered delivery service participate in the membership repair. They run a membership repair algorithm built specifically for the InterGroup system, the receiver membership repair algorithm (RMRA).

The RMRA is started when a process receives a message that signals the beginning of the MRA at a process that is an active sender. The process running the RMRA halts delivery of messages to the application and sends out the timestamp of the last message delivered to the application before it stopped delivering messages. This timestamp provides the earliest logical time at which the process can begin a new membership. It then waits for a message that describes the membership change. This message is sent by an active sender and signals the completion of the MRA. The information in this message includes a list of active senders in the new membership and the logical time at which the membership begins. This process attempts to recover and order all of the messages that precede the new membership. If it succeeds in the recovery, it installs the new membership, and successfully completes the RMRA. Otherwise, it does not successfully complete the RMRA.

The combination of the MRA and RMRA allows the active senders to provide virtual synchrony information to the entire group, while keeping the number of participating processes to a minimum.

InterGroup also provides voluntary mechanisms for processes to enter and leave the sender group. A process, wishing to join the sender group, contacts a member of the sender group that serves as a *sponsor*. The sponsor sends a message to the group requesting to add this process to the sender group. The delivery of that message signifies the addition of the process to the sender group. A process wishing to leave the sender group sends a message to the group, requesting that it be removed from the sender group. The delivery of that message signifies the removal of the process from the sender group.

Detailed membership algorithms and their interface to the other modules can be found in [4].

7 Conclusion and Future Work

The goal in designing the InterGroup protocols has been to provide the application services of group communication systems in a wide-area environment with a large number of participants, prone to large latencies and frequent faults, such as the Internet. Thus, we designed an architecture that divides the system into four major components: a control information service, a reliability service, a delivery service, and a membership service. Prior research into scaling the first two of these components allowed us to focus our efforts on the delivery and membership services.

The membership protocols of existing group communication systems have traditionally limited their scalability. The InterGroup protocols take a novel ap-

proach towards enhancing the scalability of membership protocols, while maintaining consistent message ordering and delivery within groups. In most applications, only a few group members are sending messages at any one time. A cornerstone of the InterGroup approach is the recognition that the message order and reliability constraints of a group communication system can be met by keeping only the processes currently sending messages in the group membership. The membership protocols require only the sending processes to participate in expensive group-wide decisions.

We step away from the traditional approach of choosing a delivery service to provide more flexibility to the application by allowing each process to choose a delivery service independent of the other processes in the group. This approach also allows processes that cannot meet the desired system quality of service to participate in the group, using a weaker delivery service, and improves scalability of the system.

We are currently undertaking simulation studies of the various aspects of the protocols in order to determine the scalability bounds. We are also measuring the performance of our implementation in order to investigate the performance characteristics. We are expecting to release an implementation of the InterGroup protocols soon, for further testing and use by applications.

References

[1] D. A. Agarwal, L. E. Moser, P. M. Melliar-Smith, and R. K. Budhia. The Totem multiple-ring ordering and topology maintenance protocol. *ACM Transactions on Computer Systems*, 16(2):93–132, May 1998.

[2] Y. Amir, D. Dolev, S. Kramer, and D. Malki. Transis: A communication subsystem for high availability. In *Proceedings of the 22nd IEEE International Symposium on Fault-Tolerant Computing*, pages 76–84, New York, NY, July 1992.

[3] Y. Amir, L. E. Moser, P. M. Melliar-Smith, D. A. Agarwal, and P. Ciarfella. The Totem single-ring ordering and membership protocol. *ACM Transactions on Computer Systems*, 13(4):311–342, November 1995.

[4] K. Berket. *The InterGroup Protocols: Scalable Group Communication for the Internet*. PhD thesis, Department of Electrical and Computer Engineering, University of California, Santa Barbara, CA, 2000.

[5] K. P. Birman and R. Van Renesse, editors. *Reliable Distributed Computing with the Isis Toolkit*. IEEE Computer Society Press, 1994.

[6] V. G. Cerf and R. E. Kahn. A protocol for packet network intercommunication. *IEEE Transactions on Communications*, 22(5):647–648, May 1974.

[7] S. Floyd, V. Jacobson, C.-G. Liu, S. McCanne, and L. Zhang. A reliable multicast framework for light-weight sessions and application level framing. *IEEE/ACM Transactions on Networking*, 5(6):784–803, December 1997.

[8] I. Foster and C. Kesselman, editors. *The Grid, Blueprint for a New Computing Infrastructure*. Morgan Kaufmann Publishers, Inc., 1998.

[9] K. Guo and L. Rodrigues. Dynamic light-weight groups. In *Proceedings of the 17th IEEE International Conference on Distributed Computing Systems*, pages 33–42, Baltimore, Maryland, May 1997.

[10] I. Keidar, J. Sussman, K. Marzullo, and D. Dolev. A client-server oriented algorithm for virtually synchronous group membership in WANs. In *Proceedings of the 20th IEEE International Conference on Distributed Computing Systems*, pages 356–65, Taipei, Taiwan, April 2000.

[11] L. Lamport. Time, clocks, and the ordering of events in a distributed system. *Communications of the ACM*, 21(7):558–565, July 1978.

[12] L. E. Moser, Y. Amir, P. M. Melliar-Smith, and D. A. Agarwal. Extended virtual synchrony. In *Proceedings of the 14th IEEE International Conference on Distributed Computing Systems*, pages 56–65, Poznan, Poland, June 1994.

[13] L. E. Moser, P. M. Melliar-Smith, R. K. Budhia D. A. Agarwal, and C. A. Lingley-Papadopoulos. Totem: A fault-tolerant multicast group communication system. *Communications of the ACM*, 39(4):54–63, April 1996.

[14] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson. RTP: A transport protocol for real-time applications. IETF Request for Comments: 1889, January 1996.

[15] P. Sharma, D. Estrin, S. Floyd, and L. Zhang. Scalable session messages in SRM using self-configuration. Technical Report 98-670, USC, February 1998.

[16] R. van Renesse, K. Birman, M. Hayden, A. Vaysburd, and D. Karr. Building adaptive systems using ensemble. *Software: Practice and Experience*, 28(9):963–979, July 1998.

[17] R. van Renesse, K. P. Birman, and S. Maffeis. Horus: A flexible group communication system. *Communications of the ACM*, 39(4):76–83, April 1996.

[18] B. Whetten, T. Montgomery, and S. Kaplan. A high performance totally ordered protocol. In *Proceedings of the International Workshop on Theory and Practice in Distributed Systems*, pages 33–57, Dagstuhl Castle, Germany, September 1994. Springer-Verlag.

Introducing Fault-Tolerant Group Membership Into The Collaborative Computing Transport Layer

R. J. Loader†, J. S. Pascoe† and V. S. Sunderam‡

†Department of Computer Science
The University of Reading
United Kingdom
RG6 6AY
{Roger.Loader | J.S.Pascoe}@rdg.ac.uk

‡Math & Computer Science
Emory University
Atlanta, Georgia
30302
vss@mathcs.emory.edu

Abstract. In this paper we introduce the novel election based fault tolerance mechanisms recently incorporated into the Collaborative Computing Transport Layer (CCTL). CCTL offers the atomic reliable multicast facilities used in the Collaborative Computing Framework (CCF). Our approach utilizes a reliable IP multicast primitive to implement two electorial algorithms that not only form consensus, but efficiently deliver a compact matrix based view of the network. This matrix can subsequently be analyzed to identify specific network failures (e.g. partitioning). The underlying premise of the approach being that by basing fault tolerance on a reliable multicast primitive, we eliminate the need for specific keep-alive packets such as heartbeats.

1 Introduction

The *Collaborative Computing Frameworks (CCF)* [2] is a suite of software systems, communications protocols, and methodologies that enable collaborative, computer-based cooperative work. CCF constructs a virtual work environment on multiple computer systems connected over the Internet, to form a collaboratory. In this setting, participants interact with each other, simultaneously access or operate computer applications, refer to global data repositories or archives, collectively create and manipulate documents spreadsheets or artifacts, perform computational transformations and conduct a number of other activities via telepresence. CCF is an integrated framework for accomplishing most facets of collaborative work, discussion, or other group activity, as opposed to other systems (audio tools, video/document conferencing, display multiplexers, distributed computing, shared file systems, whiteboards) which address only some subset of the required functions or are oriented towards specific applications or situations. The CCF software systems are outcomes of ongoing experimental research in distributed computing and collaboration methodologies.

CCF consists of multiple coordinated infrastructural elements, each of which provides a component of the virtual collaborative environment. However, several of these subsystems are designed to be capable of independent operation. This is to exploit the benefits of software reuse in other multicast frameworks. An additional benefit is that individual components may be updated or replaced as the system evolves. In particular, CCF is built on a novel communications substrate called the *Collaborative Computing Transport Layer (CCTL)* [5] and it is this that is the focus of this paper.

V.N. Alexandrov et al. (Eds.): ICCS 2001, LNCS 2073, pp. 326–335, 2001.
©Springer-Verlag Berlin Heidelberg 2001

CCTL is the fabric upon which the entire system is built. A suite of reliable atomic communication protocols, CCTL supports sessions or heavyweight groups and channels (with relaxed virtual synchrony) that are able to exhibit multiple Qualities of Service (QoS) semantics. Unique features include a hierarchical group scheme, use of tunnel-augmented IP multicasting and a multithreaded implementation. Other novel inclusions are fast group primitives, comprehensive delivery options and signals.

The original CCTL did not incorporate failure resilience and experiences with the system demonstrated this to be a critical requisite of any group communication protocol. To address this, the reliable multicast primitive has been modified to give reports of suspected failure i.e. it has been enhanced to act as a failure detector. Thus every multicast message acts as a probe of the sessions liveness. The main advantage of the approach is that it does not require keep-alive packets (e.g. heartbeats). This not only improves bandwidth utilization, but also increases scalability and reduces latency.

An error monitor protocol was introduced to process failure reports and to appropriately signal a second error handling protocol that an election must take place to form consensus. We base consensus on a novel electorial algorithm that compiles a matrix representation of the networks state. From this, we postulate that certain matrix transformations will identify specific types of network failure.

The remainder of this paper is structured as follows. Section 2 introduces CCTL and outlines its architectural design. Section 3 focuses on the CCTL failure model before section 4 describes the lengths to which the architecture was adapted. Sections 5 and 6 describe the error monitor and error handler protocols before section 7 outlines the role of a failure log and how the votes are returned. Section 8 discusses how the result is calculated. Finally, in section 9 we give our conclusions and outline the future directions of the research.

2 An Introduction To CCTL

CCTL is the communication layer of the CCF and as such it provides channel and session abstractions to clients. At its lowest level, CCTL utilizes IP multicast whenever possible. Given the current state of the Internet, not every site is capable of IP multicast over WANs. To this extent, CCTL uses a novel tunneling technique similar to the one adopted in the MBone. At each local subnet containing a group member is a multicast relay. This multicast relay (called *mcaster*) receives a UDP feed from different subnets and multicasts it on its own subnet. A sender first multicasts a message to its own subnet, and then sends the tunneled message to remote mcasters at distant networks. The tunneled UDP messages contains a multicast address that identifies the target subnet. TCP-Reno style flow control schemes and positive acknowledgments are used for data transfer, resulting in high bandwidth as well as low latencies. This scheme has proven to be effective in the provision of fast multiway communications both on local networks and on wide area networks. IP multicast (augmented by CCTL flow control and fast acknowledgment schemes) on a single LAN greatly reduces sender load, thus, throughput at each receiver is maintained near the maximum possible limit (approximately 800 kB/s on Ethernet) with the addition of more receivers. For example, with a 20 member group, CCTL can achieve 84% of the throughput of TCP to one destination. If in this situation TCP is used, the replicated transmissions that are required by the sender cause receiver throughput to deteriorate as the number of hosts increases. A similar effect is observed for WAN's; Table 1 in [6] compares throughput to multiple receivers from one sender using TCP and CCTL.

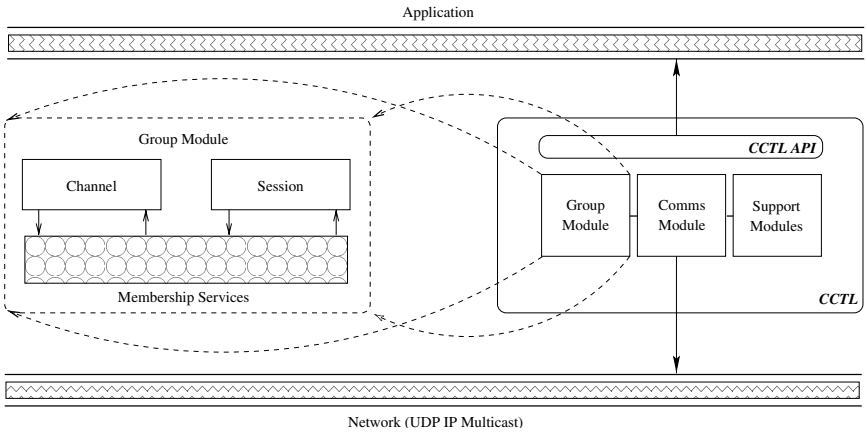


Fig. 1. CCTL Architecture

CCTL offers three types of delivery ordering: *atomic*, *FIFO* and *unordered*. FIFO ordering ensures that messages sent to process q by process p are received in the same order in which they were sent. FIFO guarantees point-to-point ordering but places no constraint on the relative order of messages sent by p and q when received by a third process r.

CCTL offers both reliable and unreliable message delivery. Reliable delivery guarantees that messages sent by a non-faulty process are eventually received (exactly once) by all non faulty processes in the destination set. In a group communication system, this can only be defined in relation to view change operations (membership protocols).

2.1 Architecture

Hierarchical Group Architecture CCTL is logically implemented as a group module, interposed between applications (clients) and the physical network. This module implements the CCTL API and provides session and channel abstractions. Recall that channels are light weight groups supporting a variety of QoS semantics. Note that related channels combine to form a heavy-weight group or session. Recall also that sessions provide an atomic virtually synchronous service called the *default channel*. Sessions and channels support the same fundamental operations (join, leave, send and receive) but many channel operations can be implemented efficiently using the default session channel. Session members may join and leave channels dynamically, but the QoS for a particular channel is fixed at creation. Channels and sessions are destroyed when the last participant leaves.

Fig. 1 shows the CCTL architecture. The group module consists of channel membership, QoS and session sub modules. The channel membership module enforces view change messages (join and leave). The QoS module also provides an interface to lower-level network protocols such as IP multicast or UDP and handles internetwork routing (IP multicast to a LAN, UDP tunneling over WANs).

Several researchers have proposed communication systems supporting light-weight groups. These systems support the dynamic mapping of many light-weight groups to

a small set of heavy-weight groups. CCTL statically binds light-weight channels to a single heavyweight session, mirroring the semantics of CSCW environments.

As noted above, CCTL implements channel membership using the default session channel. Session participants initiate a channel view change by multicasting a view change request (join or leave) on the default channel. The channel membership sub module monitors the default channel and maintains a channel table containing name, QoS and membership for all channels in the session. All session participants have consistent channel tables because view change requests are totally ordered by the default channel. This technique simplifies the execution of channel membership operations considerably. For instance, the message ordering imposed by the default channel can be used for ordering view changes. Furthermore, the implementation of channel name services is trivial, requiring a single lookup in the channel table. The architecture of CCTL logically separates channel control transmission (using the default channel) and regular data transmission. This separation increases flexibility by decoupling data service quality from membership semantics.

The hierarchical architecture is also scalable. Glade *et al.* [3] argue that the presence of recovery, name services and failure detectors degrade overall system performance as the number of groups increases. Typically failure detectors periodically poll group members[1]. CCTL performs failure detection on transmission of each multicast message. When a failed process is detected, a unified recovery procedure removes the process from all channels simultaneously, thus restoring the sessions health.

3 Failure And CCTL

Recall that CCTL is implemented as a multithreaded system. However, only the channel sender (CS) thread associated with each channel is of direct interest here; although we acknowledge that the impact of failure during a membership change operation is also an important issue. The sender thread implements a reliable multicast protocol and there is one instance of it for each channel. For each host, the sender thread provides a fault report for every send operation that fails. These operations include the *channel id*, the message sequence number and the *session ids* of those members that have not acknowledged within a time-out [2]. The *session id* of each defaulting session member is encoded as a bit mask.

It is envisaged that members can suffer a number of failures (e.g. process crash, link crash). This can result in either a complete or a *partial failure* of one or more channels. The former problem will be handled by forming a consensus amongst live session members that failed hosts should be removed from the session.The latter problem will be rectified with a second election.

The impossibility result for achieving consensus in asynchronous, message passing systems is well known. Fortunately the addition of even a weak failure detector allows protocols that solve consensus to be developed.

The comprehensive collection and presentation of the scale of the partial failure problem is an important issue. It is possible to provide automatic closing of channels if it is clear that the majority of hosts are reporting a constant stream of irregularities.

[1] E.g. Horus [7, 1] transmits a heartbeat message every two seconds.

[2] The retransmission time-out is actually the slowest round-trip latency for the hosts in the session plus an arbitrary constant

Automatic selective closing of channels will require further investigation as it involves interaction with applications. This is currently the subject of active research.

Symmetric link failures are dealt with using a seniority mechanism. If the session owner can not be reached (i.e. the network has partitioned or the host has suffered a process crash failure), then the most senior of the remaining members can assume the role of the session owner. Should the network subsequently remerge, the most senior of the session owners asserts their authority and any others revert to being standard clients. Typically, a network remerge is detected by the presence of multiple session owners.

The loss of given session member generates errors for any session member that is party to a reliable channel. All live members receive copies of messages but the sender will attempt to contact the failed member. Before the integration of fault tolerance, this resulted in degraded performance that was exasperated as the number of outstanding failures was increased.

4 Adapting The Architecture

It was desirable to investigate if fault tolerance could be introduced without major changes to the existing applications code. The additions to the architecture are given in Fig. 2. Three new threads have been added; these are called the *Error Handler (EH)*, *Error Monitor (EM)* and *Election Timer (ET)*. In addition a new UDP socket, called *FailFd*, is used to allow direct UDP communication between the channel sender, error monitor, error handler and election timer threads. The error detector and error handler also use a new reliable channel called the *fail channel*. The fail channel is similar to the default channel in that every member automatically joins it upon joining the session. Similar to the concept of the session owner one error handler will act as a coordinating master EH(M) and the rest as slaves.

4.1 Monitoring Failure By Augmenting The Sender Thread

The sender thread provides a reliable multicast over CCTL channels. Reliability is ensured by noting acknowledgments from the members of the channel. The details of channel membership are compactly included in the reliable channel message exchanges as an array of unsigned integers. The session id is used to address the bit when its value is changed. When used to represent channel membership, the mask will be termed the

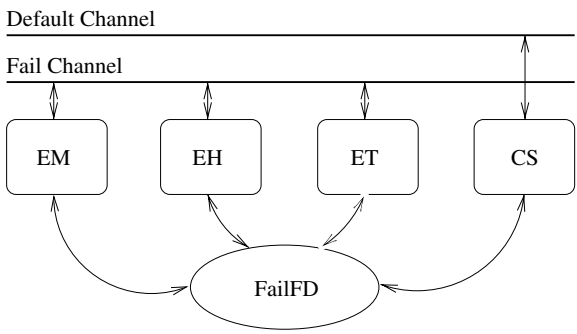


Fig. 2. The Additional Architecture

channel member mask. For the purposes of this discussion membership is indicated by a 1 and non-membership by a 0. When a message is sent, a copy of the channel membership mask is included. This is used by the sender thread to check off the destination session members as they acknowledge. Since the sender's copy is sent via shared memory it is it automatically marked as having acknowledged. In this context the copy has become an *acknowledgment mask*. The state of the acknowledgment mask after the retransmission time-out will be termed the *channel error mask*.

5 The Error Monitor Protocol

The Error Monitor thread (EM), provides first level processing of failure and failure correction reports from the transport mechanism. The main role of the EM is the maintenance of a failure log (that stores failure reports) and a *channel monitor mask* which is used to indicate the failure status of all channels. The log is a shared data structure between the Error Monitor and Error Handler threads. Details of how the log data is recorded are not currently pertinent and are deferred to section 7. It is assumed at this stage that 'add to log' and 'remove from log' functions for a single failure report are present.

Failure reports are transmitted reliably and contain the channel error mask to identify those hosts that did not acknowledge the message. Recall that in CCTL, a dedicated channel (called the fail channel) is provided for the transmission of failure reports and the operation of the election protocols. As each host joins the session, they are automatically admitted to the fail channel. Thus, a message transmitted on the fail channel will be multicast to every host in the session.

On the detection of a failure, the reliable multicast transport mechanism will attempt to resend the message. Each new failure report results in the log being updated. If a failure report is received for a message that has not been logged, then a new entry is made in the log. Otherwise, the appropriate log entry is located, and the corresponding number of failure reports is incremented. Should a host recover and subsequently acknowledge an outstanding message, then the associated number of failure reports is decremented. If the number of reports reaches 0, then the entry is pruned from the log. If the log becomes empty, then an ER_CLEAR message is sent to the Error Handler.

If the number of failure reports exceeds a confirmation threshold[3] for any failure report, the protocol sends an ER_IND message to the Error Handler thread (EH) to signal that a confirmed failure has occurred. Thus, the complete protocol can be described as:

1. A FAIL_REP message results in the log being scanned. If the FAIL_REP corresponds to a known failure, then the number of failure reports associated with that record is incremented. If this value exceeds the confirmation threshold, then this indicates a confirmed failure and an ER_IND message is sent to the EH. Otherwise, a new entry is added to the log and the channel monitor mask is updated to indicate the fault status of the channel.
2. A FAIL_CORR message decrements the number of failure reports associated with a message. If all of the failing hosts subsequently deliver FAIL_CORR messages for a report, then the corresponding entry is pruned from the log. If the log becomes empty then an ER_CLEAR message is sent to the Error Handler.

[3] The value used in our implementation of the approach is 3.

3. The reception of an EL_START message from the EH means that any new FAIL_REP (or FAIL_CORR) messages are suppressed, except for those that result from the EL_PROBE and EL_CALL messages (see below). The occurrence of these events must also generate an ER_IND.
4. An EL_RESULT message contains details of which members are to be removed from the session. In this case, the log is pruned of all entries relating to those failed members.
5. Finally, an EL_END message indicates that the process is complete.

A further role of the EM thread is to govern an election time-out during the voting phase. This topic will be explored in the next section.

6 The Error Handler Protocol

The function of the EH thread is to execute the election based failure recovery protocol. As noted above, there are two versions of the EH protocol; the *master*, referred to as EH(M) and the *slaves* EH(S). Clearly, EH(S) is a subset of EH(M). The master and slaves collectively operate the protocol to request, call and vote in two elections to remove failing hosts and deal with partial failures. Either the EH(M) or any EH(S) can request an election as a consequence of receiving an ER_IND message on the fail channel from its local EM thread[4]. Thereafter the protocol proceeds as follows:

1. The host requesting an election sends a multicast EL_REQ to the EH(M). This also informs the other session members that an election is to take place.
2. The EH(M) responds to an EL_REQ by setting the number of expected failures to zero and multicasting an EL_START followed by an EL_CALL. If there is an ER_IND generated as a consequence of the call then the number of expected failures can be calculated from the channel error mask. A time-out on the voting phase is set.
3. All recipients of the EL_CALL send a multicast EL_PROBE on the fail channel. This can result in fresh failure reports being generated, the purpose of which is for all of the sessions participants to obtain an up-to-date view of the sessions liveness.
4. An EL_PROBE message can generate an ER_IND message from the EM thread because of existing and possibly new failures. Alternatively an EL_PROBE_SUCC message is sent by the EH(M), which indicates that all hosts are capable of responding on the fail channel and that problems previously reported have been resolved.
5. Regardless of whether an ER_IND is received, an EL_RETURN is sent to the error master using a point-to-point message. The return consists of the latest channel error mask plus a serialized form of the session member's failure log.
6. The EH(M) receives the EL_RETURN from each live host. The count is terminated either by all expected returns being received or the EM signals a time-out. The latter is required to guarantee termination when more failures have occurred since the EL_CALL. Note also, that each returned failure log is combined to form the *global failure log*.
7. The EH(M) calculates the result of the election from the data returned (see section 8). The election result is then multicast to the live session members in the form of an EL_RESULT message. Following this, each live member removes those members that are agreed to have failed from its channels.

[4] If at any time, the EH(M) is uncontactable or fails, the role is transferred to the sessions most senior live member and the election is restarted (if one is in progress). Note that the topics of EH(M) failure and its extension to partitioning are covered in more depth in [4].

8. Note that while the election is in progress, further ER_IND messages (other than those expected during the election) are queued by all Error Handlers. On completion of the election, these queues are examined and a session election is called if new failures have occurred.
9. The session election interrogates the global failure log compiled during the first election. Failures on individual channels are considered to ascertain a deeper view of the sessions health. For example, intermittent failures may be evident for a host across the entire session (i.e. for all channels). In this case, the problem can be resolved by transmitting a second EL_RESULT message. Alternatively, a host may be experiencing problems on a subset of the sessions channels. This may be resolved by resynchronizing the affected part of the system.
10. When the algorithm has finished, an EL_END message is multicast to all of the sessions participants informing them that the process is complete.

A more formal definition of the protocols expressed in state event table form is given in [4]. This defines the state variables, the necessary predicates for guarding the atomic actions and gives details of the message formats.

7 The Failure Log And Returning The Vote

The failure log records the stream of failure reports about messages on channels which have not been acknowledged in time. Suppose that the first such report on a channel arrives. Three pieces of information are given: the channel id, the message sequence number and channel error mask with bits set corresponding to the destinations that have not acknowledged the message. If the underlying condition that caused the generation of the message is not cleared, the reliable multicast will send the message again to the tardy hosts until the number of failure reports exceeds the confirmation threshold. Fault reports from the same channel, but with greater message sequence numbers, can also arrive. This is a boon since every failed trial at sending obtains the latest information about the failed hosts in the channel.

It is necessary to keep only the latest copy of the error mask for the channel because it represents the latest information on the session state as far as this channel is concerned. All message sequence numbers are recorded. Suppose that the hosts concerned are only temporarily slow in responding. A stream of FAIL_CORR messages with channel id and sequence numbers will be received. The receipt of a FAIL_CORR on a given channel decrements the total of failure reports associated with a given report. If this count reaches zero then the entry is pruned from the log. If the log is now empty, then the session is considered to be nominal.

8 Producing The Result

The purpose of the election is to assemble from the live members an up to date collective view of the health of the entire session. An initial vote can be taken on the first part of each return, namely the channel error mask resulting from the transmission of the EL_PROBE. This is called the *membership removal election* and it results in any agreed failed members being removed from the session. In the presence of partial failures, this may not repair the session and so the global failure log is then inspected.

Algorithm 1: Membership Removal Algorithm
Code for host EH(M)

Initially $V_M^Y \leftarrow V_M^N \leftarrow V_T \leftarrow n_{ret} \leftarrow 0; V_R \leftarrow false; N_s \leftarrow$ *number in session*

1: while ($n_{ret} < N_s$ - number of estimated failures (from initial ER_IND))
  2: (Receive a time-out; goto 8) $\vee$ (Receive EL_RETURN$_i$ from host i)
  3: Convert channel error mask$_i$ to V_T and add failure log$_i$ to global failure log
    4: for (j $\leftarrow$ 0; j $< N_s$; j++)
      5: if $V_T[j] = 1$ then $V_M^Y[j] \leftarrow V_M^Y[j] + 1$
      6: else $V_M^N[j] \leftarrow V_M^N[j] + 1$
  7: nret $\leftarrow$ nret + 1
8: for (j $\leftarrow$ 0; j $< N_s$; j++)
  9: if $V_M^Y[j] \geq \lceil \frac{N_s}{2} \rceil$ then $V_R[j] \leftarrow$ true
  10: else $V_R[j] \leftarrow$ false
11: Send V_R as an EL_RESULT to all hosts

Fig. 3. Membership Removal Algorithm

8.1 Membership Removal Election

Once all of the votes have been returned the EH(M) conducts the membership removal election. Declaring a vector V_M^Y for the 'yes' votes, a vector V_M^N for the 'no' votes, a temporary vector V_T, a boolean result vector V_R and an integer n_{ret} to count the number of returns, the informal counting algorithm is given in fig. 3.

The approach taken aims to build a degree of fault tolerance that can allow a minority of simultaneous failures. Suppose there are N_s current session members and n_f failures when the probe took place. The elementary case is defined to be when the group of remaining live members, $N_s - n_f$, is the majority and there are no more failures during the election. In this instance, the value of N_s is reduced by n_f. Now consider that $N_s - n_f$ is still the majority but the vote is split between those voting yes, n_{yes}, those voting no, n_{no}, and those that have failed after the probe but before they can return a result, n_{fail}. Again, the simple case is when the majority of N_s vote yes. A vote where not enough members vote yes, $n_{no} > n_{yes}$, can arise because of a time skew between members and is resolved by holding a session election. The case where not enough votes are cast $n_{yes} + n_{no} < \lceil \frac{N_s}{2} \rceil$, can occur either when an election has taken place in a minority partition or too many extra members have failed during the election. This case is regarded as unrecoverable as is the one where a minority remains.

8.2 Session Election

The execution of the first election may result in the removal of some members. In the presence of partial failures, this may not rectify the problem. An election is now performed using the global failure log to produce an overall picture of the sessions state.

The result of the election is a pair of matrices, where the row index is the channel id, c, and the column index is the host id s. A positive element in the 'yes' matrix signifies that there have been reported errors about the respective member on the corresponding channel. A zero value signifies no reported errors and a negative element indicates that s is not a member of c. Declaring a pair of two dimensional vectors V_S^Y and V_S^N to store the result, we present the session election algorithm in fig. 4.

Algorithm 2: Session Election Algorithm
Code for host EH(M)

Initially $V_S^Y \leftarrow V_S^N \leftarrow 0; V_R \leftarrow false; N_s \leftarrow$ *number in session*

1: for (c $\leftarrow$ 0; c $<$ number of channels; c++)
  2: for (s $\leftarrow$ 0; s $< N_s$; s++)
    3: if (s is a member of c) $\wedge$ (global log shows errors reported about s on channel c)
      then $V_S^Y[s,c] \leftarrow V_S^Y[s,c] + 1$
    4: else if (s is a member of c) $\wedge\neg$(global log shows errors reported about s for channel c)
      then $V_S^N[s,c] \leftarrow V_S^N[s,c] + 1$
    5: else $V_S^Y[s,c] \leftarrow V_S^N[s,c] \leftarrow -1$
6: for (s $\leftarrow$ 0; s $< N_s$; s++)
  7: if $\forall c \in channel \bullet V_S^Y[s,c] \geq \lceil \frac{N_s}{2} \rceil$ then $V_R[s] \leftarrow$ true
  8: else if $\exists c \in channel \bullet V_S^Y[s,c] \geq \lceil \frac{N_s}{2} \rceil$ then $resynchronize_channel(c)$
9: if (V_R has been updated) then send V_R as an EL_RESULT to all hosts

Fig. 4. Session Election Algorithm

9 Conclusion

This paper describes the novel fault tolerant group membership mechanisms recently incorporated into the Collaborative Computing Transport Layer.

Future work is considering a number of avenues. In the short term we are considering the migration of this approach to wireless environments. It is the opinion of the authors that collaborative computing can benefit greatly from the recent advances in wireless networks and hand-held / wearable computing. When the approach documented here has been fully evaluated, insight will be gained as to how the research should evolve.

References

1. K. P. Birman. *Building Secure and Reliable Network Applications*. Prentice Hall, 1997.
2. S. Chodrow, S. Cheung, P. Hutto, A. Krantz, P. Gray, T. Goddard, I. Rhee, and V. Sunderam. CCF: A Collaborative Computing Frameworks. In *IEEE Internet Computing*, January / February 2000.
3. B. Glade, K. P. Birman, R. Cooper, and R. Renesse. Light weight process groups in the isis system. *Distributed Systems Engineering*, 1:29–36.
4. R. J. Loader, J. S. Pascoe, and V. S. Sunderam. An Electorial Approach to Fault-Tolerance in Multicast Networks. Technical Report RUCS/2000/TR/011/A, The University of Reading, Department of Computer Science, 2000.
5. I. Rhee, S. Cheung, P. Hutto, A. Krantz, and V. Sunderam. Group Communication Support for Distributed Collaboration Systems. In *Proc. Cluster Computing: Networks, Software Tools and Applications*, December 1998.
6. I. Rhee, S. Cheung, P. Hutto, and V. Sunderam. Group Communication Support for Distributed Multimedia And CSCW Systems. In *Proc. 17th International Conference On Distributed Systems*, May 1997.
7. R. van Renesse, K. P. Birman, and S. Maffeis. Horus, a flexible group communication system. In *Communications of the ACM*, April 1996.

A Modular Collaborative Parallel CFD Workbench

Kwai L. Wong[1] and A. Jerry Baker[2]

[1] Joint Institute for Computational Science, University of Tennessee, Knoxville, TN 37996, USA
[2] Engineering Science Program, MAE&ES, University of Tennessee, Knoxville, TN 37996, USA

Abstract. Simulation of physical phenomena on computers has joined engineering mechanics theory and laboratory experimentation as the third method of engineering analysis design. It is in fact the only feasible method for analyzing many critically important phenomena, e.g., quenching, heat treating, full scale phenomena, etc. With the rapid maturation of inexpensive parallel computing technology, and high performance communications, there will emerge shortly a totally new, highly interactive computing/simulation environment supporting engineering design optimization. This environment will exist on Internet employing interoperable software/hardware infrastructures emergent today. A key element of this will involve development of computational software enabling utilization of all HPCC advances. This paper introduces the concept and details the development of a user-adapted computational simulation software platform prototype on the Internet.

1 Introduction

Simulation of physical phenomena on computers has joined engineering experiment and theory as the third method of scientific investigation. It is in fact the only feasible method for analyzing many different types of critically important phenomena, e.g., geological time scale evolution, options for clean/efficient combustion, quenching/heat treating, optimized ventilation design, etc.

The need to address such problems involves the collaborative work of scientists, engineers, and computer specialists. With the rapid infusion of parallel computing technology in recent years, the opportunity to create a unified problem-solving environment in computational mechanics to attack large-scale problems has emerged.

To meet the challenge, many issues have to be thought out and resolved. Foremost, an interoperable software infrastructure has to be built, which deals with issues like physics encapsulation, I/O streaming, heterogeneity, security, etc. Last fall, a team of engineering, computational and software scientists, at the University of Tennessee/Knoxville (UTK) started a collaborative inhouse research project to address the theoretical, algorithmic, and numerical implementation issues necessary to assess the potential for success of a Parallel Interoperable Computational Mechanics Simulation System (PICMSS) operating on

V.N. Alexandrov et al. (Eds.): ICCS 2001, LNCS 2073, pp. 336–344, 2001.
© Springer-Verlag Berlin Heidelberg 2001

the Internet. This theoretical focus is development of an implicit finite element multi-dimensional CFD software platform capable of admitting diverse CFD formulations, as well as closure models for physics phenomena, as required/derived by users.

Imagine that engineers in industry and/or academic researchers will be enabled to:

1. Remotely describe their problem to a genuinely production simulation environment from their local PC,
2. Submit their computational tasks to a pool of collaborative computers located anywhere in the country,
3. Monitor their computational tasks interactively, then,
4. Examine their final results locally from remotely available graphics utilities.

This is exactly the environment that PICMSS aims to provide. To deliver such capabilities, this project will integrate innovative software design, computational mechanics theory and parallel CS packages with proven methodologies [1,4].

2 Objective

The objective is to develop, advance and proliferate a Parallel Interoperable Computational Mechanics Simulation System (PICMSS). The common lament in the engineering computational mechanics community is the lack of adaptability in available software for effectively and efficiently attacking multi-disciplinary problems in engineering design.

In constructing PICMSS, the goal is to collate the significant recent advances in the communication and information sciences, and in computational mechanics theory/practice, and to deploy the result into the hands of both researchers and engineering practitioners to better facilitate rapid simulation of engineered systems involving non-linearly coupled fluid/thermal mechanics disciplines. PICMSS will as well be designed as a faculty toolkit to support academic instruction/research in the computational mechanics of continua with operation/dissemination via Internet.

The specific goal is to develop and deliver as a national/international resource:

1. A modular parallel compute engine using, but not limited to, the finite element (FE) implementation of a weak form to computationally formulate the physics, chemistry and mathematics description of the problem statement at hand,
2. An equation-based graphical client user interface to encapsulate the FE computational theory for this differential/algebraic equation system describing the multi-disciplinary problem to be simulated,
3. A middle server layer adapted to NetSolve [2] to steer and monitor the computational state of the defined simulation, and
4. Enhancements to NetSolve to facilitate data exchange between the computing engine and users or secondary storage brokers during and after execution.

3 Scope of PICMSS

The simulator will be assembled at six levels of abstraction:

1. Construction of a JAVA graphical user interface (GUI) uniquely tailored to suit the implementation of the problem framework using language constructions familiar to engineers and researchers,
2. Conversion of this differential/closure equation system into a computational form using state-of-the-art numerics,
3. Creation of backend server interface to control and distribute computing information to processing units,
4. Expansion of the computational engine to admit various classes of problems,
5. Induction of PICMSS to NetSolve for metacomputing,
6. Exposition of output data and monitoring and control during execution.

Figure 1 graphs schematically the three-tier structural units of the PICMSS computational simulator. A client-server model is used to provide the interfaces between the users and the meta-computing engines. The PICMSS translation from GUI differential equation system to computational form will be based on, but not limited to, a finite element discrete implementation of a weak form. An equation editor will be developed for users to express their problem statements, constituted of non-linearly coupled partial differential, algebraic and algebraic-differential equations.

The language will be vector calculus, using the "textbook" form familiar to engineers. Geometrical (definition, discretization, etc.) inputs and ini-

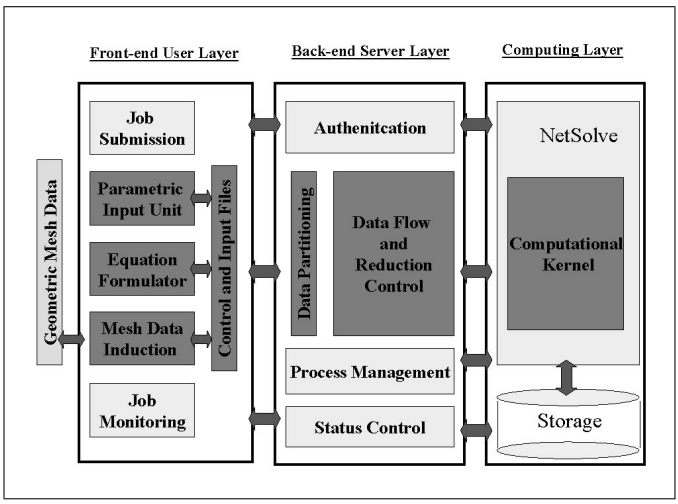


Fig. 1. Three-Tier Structural Units of PICMSS.

tial/boundary conditions will be admitted to PICMSS as external files created

at the client site. The encapsulated information will pass through a JAVA server deamon to manage and schedule resources for the computational engine to act on. The computational kernel of the simulator can be located independently on a pool of local machines, or be put under the control of NetSolve for global access. Output is piped to designated resources for local retrieval or remote examination.

The Front-end Client Interface will allow users to define, formulate and monitor their problems. The back-end Java server is designed to provide basic services for security, resource location, resource management, status monitoring, and data movement that are required for high performance computing in distributive environments. The computational kernel accepts the input file and executes the sequence of tasks defined in the control file. It is built on modular differential and integral operators acting on operands (state variables).

The encapsulation of mesh construction and assembly procedures increases the modularity and portability of the method to attack problems of different nature and formulations. It is also inherently parallel. The "solving" of partial differential equations generally involves a time integration scheme, a Newton or quasi-Newton iteration strategy for nonlinear corrections, and a linear solver for a large scale matrix statements. The use of Krylov iterative methods [1,4] is primarily coupled with preconditioners to improve convergence as the problem size increases.

To insure that PICMSS is well integrated into the Computational Grid movement that is now emerging, it will utilize NetSolve, which has emerged as a leading software environment for building grid-enabled PSE's [3]. NetSolve (www.cs.utk.edu/netsolve/) is a software environment for networked computing that transforms disparate computers and software libraries into a unified, easy-to-access computational service.

4 Benchmarking PICMSS

Benchmark problems to verify the integrity, validity, and scalabilty of PICMSS include three-dimensional flows in a straight channel, a lid-driven cavity, and a thermally-driven cavity. Exact solutions for the velocity profiles of the fully developed duct flow are available in two and three dimensions [5]. Data of computational benchmarks are also available for the three dimensional driven and thermal cavities. Using the velocity-vorticity formulation for the incompressible Navier Stokes equations, a collection of solutions [6] for a range of different Reynolds numbers and Rayleigh numbers were developed under a predecessor effort on various parallel machines. Herein, the computed results for benchmark 2D and 3D duct flow are presented using both the velocity-vorticity incompressible Navier-Stokes (INS) formulation and the pressure projection INS formulation.

4.1 Governing Equations

The nondimensional, laminar flow incompressible Navier-Stokes equation system with Boussinesq body-force approximation in primitive form is,

Continuity:

$$\nabla \cdot \mathbf{u} = 0, \tag{1}$$

Momentum:

$$\frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u} \cdot \nabla)\mathbf{u} = -\nabla P + \frac{1}{\mathrm{Re}}\nabla^2 \mathbf{u} - \frac{\mathrm{Gr}}{\mathrm{Re}^2}\Theta \hat{g}, \tag{2}$$

Energy:

$$\frac{\partial \Theta}{\partial t} + (\mathbf{u} \cdot \nabla)\Theta = \frac{1}{\mathrm{RePr}}\nabla^2 \Theta, \tag{3}$$

where $\mathbf{u} = \mathbf{u}(\mathbf{x}, t) = (u, v, w)$ is the velocity vector field (resolution), t is the time, $\mathbf{x} = (x, y, z)$ is the spatial coordinate, Θ is the potential temperature, $\hat{\mathbf{g}}$ is the gravity unit vector and P is the kinematic pressure. The nondimensional parameters are Reynolds number (Re), Prandtl number (Pr), and Grashof number (Gr) defined as,

$$Re = \frac{U_r L_r}{\nu} \quad , \quad Pr = \frac{\nu}{\alpha} \quad , \quad Gr = \frac{\beta g \Delta T_r L_r^3}{\nu^2},$$

where L_r and U_r are the reference length and velocity respectively, g is the gravity acceleration, ν is the kinematic viscosity, β is the coefficient of volume expansion, α is the thermal diffusivity, and ΔT_r is the reference temperature difference. The vorticity vector, $\mathbf{\Omega} = (\Omega_x, \Omega_y, \Omega_z)$, is kinematically defined as

$$\mathbf{\Omega} = \nabla \times \mathbf{u}. \tag{4}$$

Taking the curl of definition (4), together with the incompressibility constraint (1) and the vector identity,

$$\nabla \times \mathbf{\Omega} = \nabla \times \nabla \times \mathbf{u} = \nabla(\nabla \cdot \mathbf{u}) - \nabla^2 \mathbf{u}, \tag{5}$$

yields the velocity vector Poisson equation system

$$\nabla^2 \mathbf{u} = -\nabla \times \mathbf{\Omega}. \tag{6}$$

Taking the curl of the momentum equation (2) eliminates any gradient field. Applying equation (1) and noting $\nabla \cdot \mathbf{\Omega} = 0$, the vorticity transport equation is

$$\frac{\partial \mathbf{\Omega}}{\partial t} + (\mathbf{u} \cdot \nabla)\mathbf{\Omega} - (\mathbf{\Omega} \cdot \nabla)\mathbf{u} = \frac{1}{\mathrm{Re}}\nabla^2 \mathbf{\Omega} - \frac{\mathrm{Gr}}{\mathrm{Re}^2}\nabla \times \Theta \hat{\mathbf{g}}. \tag{7}$$

Hence, the velocity-vorticity formulation for the laminar INS equations system with Boussinesq approximation in three dimensions can be written as

$$\nabla^2 \mathbf{u} + \nabla \times \mathbf{\Omega} = 0, \tag{8}$$

$$\frac{\partial \mathbf{\Omega}}{\partial t} + (\mathbf{u} \cdot \nabla)\mathbf{\Omega} - (\mathbf{\Omega} \cdot \nabla)\mathbf{u} - \frac{1}{\mathrm{Re}}\nabla^2 \mathbf{\Omega} + \frac{\mathrm{Gr}}{\mathrm{Re}^2}\nabla \times \Theta \hat{\mathbf{g}} = 0, \tag{9}$$

$$\frac{\partial \Theta}{\partial t} + (\mathbf{u} \cdot \nabla)\Theta - \frac{1}{\mathrm{RePr}}\nabla^2 \Theta = 0. \tag{10}$$

This formulation consists of three velocity Poisson equations that couple the velocity and vorticity components kinematically via the constraint of continuity, three vorticity transport equations that describe flow kinetics, and the transport equation for temperature. The problem is well posed even though only $\mathbf{u}$ is specified at the solid wall. However, to successfully solve the momentum equations, a boundary condition for the vorticity vector on the no-slip wall must be specified. The boundary condition dervied from equation (6) to constrain the component of the equation $\mathbf{\Omega} = \nabla \times \mathbf{u}$ normal to S. Hence, the vorticity at a no-slip wall is represented by

$$\mathbf{\Omega}_{wall} = \hat{\mathbf{n}} \cdot \nabla_S \times \mathbf{u} \tag{11}$$

4.2 Problem Translation in PICMSS

The approach to numerics theory expression generalization employs a discrete implementation of a weak form for INS conservation law systems. The weak form is a collection of theory-rich concepts and techniques for the construction of a near-optimal approximate solution processes for the initial-boundary value problem statements presented in computational continuum mechanics.

A truly valuable instruction distinction accrues to use of calculus in weak form implementation. Specifically, in finite element form, all resultant algorithm components are analytically formed (no Taylor series) and are universally expressed as a matrix statement, with all formation processes conducted on a master element domain. Such weak form constructions are captured and formulated to describe the model problems by choosing the appropriate operators and operands defined in the PDE system. Examples of such operators are

$$\nabla^2(?) \ , \ \nabla \times (?) \ , \ \nabla \cdot (?) \ , \ \frac{\partial(?)}{\partial t} \ , \ \{(?) \cdot \nabla\}(?) \tag{12}$$

where (?) can be any operand or defined variables of the PDE system. Only one operand (right-hand side) is needed for linear operators. An additional left-hand side operand is required to implement the nonlinear operators.

The finite element weak form approximation of the state variable is,

$$\mathbf{q}(\mathbf{x}, t) \approx \mathbf{q}^h \equiv \sum_{j=1}^{m} N_j(\mathbf{x})^t \mathbf{Q}_j(t), \tag{13}$$

where $\mathbf{q}$ represent state variables $\{\mathbf{u}, \mathbf{\Omega}, \mathbf{\Theta}, \mathbf{P}\}$ at nodes of basis functions $\{N\}$. The operators are then represented by element-rank matrices of the following forms,

$$\nabla^2 = \int_{\sigma_e} \nabla\{N\} \cdot \nabla\{N\}^T dV, \tag{14}$$

$$\frac{\partial}{\partial x} = \int_{\sigma_e} \{N\} \frac{\partial \{N\}^T}{\partial x} dV, \tag{15}$$

$$\frac{\partial}{\partial y} = \int_{\sigma_e} \{N\} \frac{\partial \{N\}^T}{\partial y} dV, \tag{16}$$

$$\frac{\partial}{\partial z} = \int_{\sigma_e} \{N\} \frac{\partial \{N\}^T}{\partial z} dV, \tag{17}$$

$$\frac{\partial \{Q\}}{\partial t} = \int_{\sigma_e} \{N\}\{N\}^T dV \{Q\}, \tag{18}$$

The nonlinear terms, $\{(?) \cdot \nabla\}(?)$, will consist of a combination of the following integrals,

$$\{?\}\frac{\partial}{\partial x} = \{?\} \int_{\sigma_e} \{N\} \frac{\partial \{N\}}{\partial x} \{N\}^T dV, \tag{19}$$

$$\{?\}\frac{\partial}{\partial y} = \{?\} \int_{\sigma_e} \{N\} \frac{\partial \{N\}}{\partial y} \{N\}^T dV, \tag{20}$$

$$\{?\}\frac{\partial}{\partial z} = \{?\} \int_{\sigma_e} \{N\} \frac{\partial \{N\}}{\partial z} \{N\}^T dV. \tag{21}$$

PICMSS adapts this methodology to prescribe the PDE which models the physics of a problem supplied by the user.

4.3 Results for Channel Flows

Fully developed, and developing flow in a straight rectangular channel tests validity and accuracy of the algorithm boundary conditions and the constraint of continuity. The verification steady state fully-developed axial (u)-velocity distribution is [5]

$$u = \frac{48}{\pi^3} \frac{\xi_u(y,z,h)}{\varphi} \tag{22}$$

$$\xi_u(y,z,h) = \sum_{n=1,3,5}^{N} (-1)^{\frac{n-1}{2}} \left[1 - \frac{\cosh(n\pi y/2h)}{\cosh(n\pi/2)}\right] \frac{\cos(n\pi z/2h)}{n^3} \tag{23}$$

$$\varphi = 1 - \frac{192}{\pi^5} \sum_{n=1,3,5}^{N} \frac{\tanh(n\pi/2)}{n^5} \tag{24}$$

where h is the duct half-height and N is a large integer, e.g, $N = 200$ is used.

Solutions of fully developed straight channel flows in 2D and 3D were established for timing and verified with exact solutions. The GMRES Krylov sparse iterative solver, in conjunction with the least square preconditioner, was selected to solve the matrix statement. Computation were carried out on the Compaq supercluster at the Oak Ridge National Laboratory and a PC Cluster with Gigabit Ethernet interconnect at the University of Tennessee, Knoxville. Results of timing analyses for the 2D case of mesh size M=45x89 and the 3D case of mesh size M=15x15x51 using the pressure projection INS formulation are shown in Tables 1 and Table 2. Timing for the 3D case of mesh size M=24x24x25 using the velocity-vorticity formualtion is shown in Tables 3.

Superlinear speedup is observed in Table 3. Such phenomenon is not unusual for the treatment of large sparse matrices. It is primarily due to the result of more efficient utilization of cache on local processors. The selected problem size is too large to be solved using three processors.

Table 1. : Timing Analyses, Steady 2D Straight Channel Flow, M=45x89, Pressure Projection INS Formulation

Machine Platform	Number of Processors	Last Time Step Assemble(sec)	Last Solve(sec)	Total time(sec)	Speed Up
Compaq	3	2.374	1.463	244. 2	1.00
Compaq	5	0.957	1.058	138.9	1.758
Compaq	15	0.190	.992	80.2	3.044
PC	3	5.73	8.1	858.0	1.00
PC	5	2.428	5.91	517.0	1.66
PC	15	0.510	2.73	220.0	3.9

Table 2. : Timing Analyses, Steady 3D Straight Channel Flow, M=15x15x51, Pressure Projection INS Formulation

Machine Platform	Number of Processors	Last Time Step Assemble(sec)	Last Solve(sec)	Total time(sec)
Compaq	15	9.8726	1.555	754.3
PC	15	24.96	8.14	1980.5

5 Conclusions and Future Work

An interoperable parallel computational platform, PICMSS, has been designed to facilitate rapid simulation of engineering systems involving non-linear coupled fluid/thermal processes. It serves as a collocation point for incorporating novel computational methodologies as well as state-of-the-art advances in communication and information sciences. Expansion to assimilate new and innovative algorithm constructions is intrinsic to the design of PICMSS. Installation

Table 3. : Timing Analyses, Steady 3D Straight Channel Flow, M=24x24x24, Vorticity-Velocity Formulation

Machine Platform	Number of Processors	Last Time Step Assemble(sec)	Last Solve(sec)	Total time(sec)	Speed Up
Compaq	3	1034.4	28.1	11722.2	1.0
Compaq	6	234.6	15.58	2776.9	4.2
Compaq	12	78.63	5.77	944.5	12.4
Compaq	24	31.27	3.00	395.3	29.6

of PICMSS on the Internet metacomputing environment will be accomplished via the NetSolve system, which will be enhanced and adapted to manage and monitor computational resources for the proposed PICMSS environment.

The future goal is to advance and proliferate the Parallel Interoperable Computational Mechanics Simulation System capable of solving multi-disciplinary problems in engineering design. Upon establishing the first release of PICMSS, an Alliance effort is proposed to support this Internet-based computational environment. Users will be encouraged to expand the physical and numerical capacities of PICMSS as a national software resource.

6 Acknowledgments

The authors would like to acknowledge the support of computer resources from the Computer Science and Mathematics Division and the Solid State Division at Oak Ridge National Laboratory.

References

[1] Balay, S., Gropp, W. D., McInnes, L. C., and Smith, B. F., The Portable, Extensible Toolkit for Scientific Computing, Version 2.0.24, Argonne National Laboratory, 1999.
[2] Casanova, H., Dongarra, J. and Seymour, K., Users' Guide to NetSolve, Department of Computer Science, University of Tennessee, 1998.
[3] Foster, I. and Kesselman, C. eds, The Grid: Blueprint for a New Computing Infrastructure, San Francisco, Morgan Kaufman Publishers, 1999.
[4] Hutchinson, S., Prevost, L., Shadid. J., and Tuminaro, R., Aztec Users' Guide, Version 2.0 Beta, Massively Parallel Computing Research Laboratory, Sandia National Laboratory, 1998.
[5] White, T., Viscous Fluid Flow, New York, McGraw-Hill, 1974.
[6] Wong, K. L., A Parallel Finite Element Algorithm for 3D Incompressible Flow in Velocity-Vorticity Form, PhD Dissertation, The University of Tennessee, 1995.

Distributed Name Service in Harness

Tomasz Tyrakowski†, Vaidy Sunderam† and Mauro Migliardi†

†Department of Math & Computer Science
Emory University
Atlanta, Georgia
30302
{ttomek | vss | om}@mathcs.emory.edu

Abstract. The Harness metacomputing framework is a reliable and flexible environment for distributed computing. A shortcoming of the system is that services are dependent on a name service (a single point of failure) where all Harness Distributed Virtual Machines are registered. Thus, there is a need to design and implement a more reliable name service. This paper describes the Harness Distributed Name Service (HDNS) which aims to address this shortcoming. Section 2 outlines the role of the name service in Harness. Section 3 extends this discussion by describing the design of the HDNS. Finally, in sections 4 and 5, we present the services fault-tolerance mechanisms and give our conclusions.

1 Introduction

Harness is an experimental metacomputing framework that is based upon the principle of dynamically reconfigurable, networked virtual machines. Harness supports reconfiguration not only in terms of the computers and networks that comprise the virtual machine, but also in the capabilities of the virtual machine itself. These characteristics may be modified under user control via a *plugin* mechanism, which is the central feature of the system. The plugin model provides a virtual machine environment that can dynamically adapt to meet an applications needs, rather than forcing the application to conform to a fixed model [2].

The fundamental abstraction in the Harness metacomputing framework is the *Distributed Virtual Machine* (DVM). Heterogeneous computational resources may enroll in a DVM at any time, however, at this level the DVM is not ready to accept requests from users. To begin to interact with users and applications, the heterogeneous computational resources enrolled in a DVM need to load plugins. A plugin is a software component that implements a specific service. Users may reconfigure the DVM at any time, both in terms of the computational resources enrolled and in terms of the services available by loading and unloading plugins [1].

2 Name Service In Harness

Each Distributed Virtual Machine (DVM) in Harness consists of exactly one DVM server and an arbitrary number of Harness kernels. If a DVM server crashes, it is automatically restored by one of the kernels. The DVM server is responsible for event propagation and DVM state updates - it is the center of the star (see also [1]) while the kernels can be considered as the branch nodes. New kernels may join a DVM at any

V.N. Alexandrov et al. (Eds.): ICCS 2001, LNCS 2073, pp. 345–354, 2001.
©Springer-Verlag Berlin Heidelberg 2001

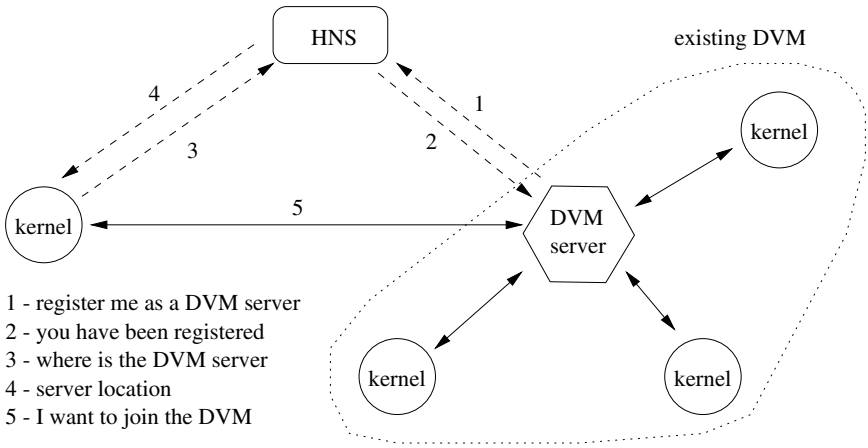


Fig. 1. A New Kernel Joining An Existing DVM.

time, performing the join protocol with the DVM server. The fact that a DVM server is automatically restored is a boon, but it also introduces some complications. New kernels added to the DVM have no mechanism for identifying where the current DVM server is, unless they consult the *Harness Name Service* (HNS) for this information. This service stores pairs of form:

(DVM name, (host, port))

where DVM name is a string unique within the naming service and (host, port) is the location of the server for the specified DVM.

The location of the name server is defined in the *harness.defaults* configuration file, which is present on all machines enrolled in the DVM. Therefore when a new kernel starts, it reads the name server location from the configuration file and then asks the name server for the location of the DVM server. If there is no server registered for the DVM, the kernel spawns a new one.

This strategy implies a specific behavior in the name service. When a DVM server attempts to register itself as a server for a DVM, the name server has to return one of the following:

- registration was successful - assume the role of DVM
- there is another DVM server registered already - the server is (host,port).

In no situation, can a name server refuse the registration of a DVM server, without giving information about the existing DVM server. The name service is also responsible for keeping the list of dormant daemons running, so that any participant can ask about the list of hosts, which can be enrolled into a DVM remotely. A dormant daemon is basically a small Java application, which listens on a specified port and spawns a Harness kernel for a particular DVM when a request comes from the network. The HNS maintains a list of the hosts dormant daemons run on. A dormant daemon also registers itself in the name server after it starts, but this scenario is much simpler - the name server just registers it and inserts the details into its store.

Both DVM servers and dormant daemons are required to refresh their entries periodically, otherwise the name server removes the corresponding entry from its tables. In summary, there are the following types of requests:

- register a DVM server (or return the information about the current server)
- register a dormant daemon
- refresh a DVM server entry (which is equal to re-registering it)
- refresh a dormant daemon entry (equivalent to re-registering)
- give information about the DVM server for a specified DVM
- give a list of hosts running dormant daemons

As was noted above, the name service is essential to the consistent operation of DVM servers and kernels. Thus, a single name server is a single point of failure for the whole DVM. When the name server crashes or is unavailable, new kernels are not able to enroll into a DVM and in the case of a DVM server crash, there is no way to restore it[1]. The name service reliability is critical in this context, and it is impossible to achieve reliability with a centralized name server operating on one physical machine. For this reason, there is a need to design and implement a distributed naming service.

3 The Harness Distributed Name Service

The *Harness Distributed Name Service* (HDNS) is intended to be more reliable provider of the services described in section 2. Initially, it has to consist of more than one physical machine, which immediately implies a number of issues. The first problem is the choice of topology. Another, is synchronization and propagation of data. The ring topology is most suited to our requirements as it is simple to implement. Instead of a single name server, we have a ring consisting of up to 8 physical machines. The number of machines is large enough to provide satisfactory reliability and fault tolerance. This should also keep the ring fast enough to handle client requests and not cause delays. As was noted above, the DVM servers and dormant daemons have to refresh their name server entries periodically and it is inefficient if this causes unnecessary delays in their work. In addition, the ring should have multiple injection points, which means that updates and lookups can be performed using any of the ring members. This is not simple to achieve, although with some additional mechanisms it is possible.

The ring is directed, which means the requests go one way - only acknowledgements are allowed to traverse the ring in the opposite direction. Moreover, one channel is for communication between the servers, and a second one acts as the means by which the clients can communicate with the name servers. Thus, each ring member listens on two ports - on one server requests arrive whilst the clients (i.e. DVM servers, dormant daemons and kernels) requests are transmitted via the second one. These are known as: *server-to-server* and *server-to-client* channels, respectively. It is noteworthy to conclude the description by highlighting that all the communications use the TCP/IP protocol and JDK socket implementation.

It was noted above, that the ring has multiple injection points. Indeed, a client can contact any of the name servers and request an update or a lookup. No matter which ring member has been selected, the client's request should always be processed in the same

[1] The Harness prototype bases its operation on multicast where available. But since multicast is not present on all systems, we assume that a point-to-point protocol is used.

Algorithm 1 Process a client's request.

1. If the request type is 'lookup', return the result from the tables and STOP.
2. If the request type is 'update', do the following:
3. Check if the update is possible considering the contents of the tables.
4. If the answer in step 3 is 'no', send the proper answer to the client and STOP.
5. If the answer is 'yes', do the following:
6. Send the request to the next ring member.
7. Wait for the request to come from the name server prior to you.
8. Send the answer to the client and STOP.

way (in other words, the result of the request must not depend on which ring member is contacted).

Below are the listed requirements, which the ring should satisfy in order to be suitable for use as a name service in Harness:

1. It has to be fault tolerant, both in terms of node crashes and link breaks
2. It has to be fast enough not to cause delays in the work of DVM servers, dormant daemons and kernels
3. All the requests listed in section 2 have to be properly processed and propagated

Unfortunately these principles contradict each other, but as it shown later, it is not possible to satisfy them all. So, in summary:

- the ring consists of up to 8 physical nodes
- the ring is directed
- it has multiple injection points
- it uses the TCP/IP protocol

The general algorithm for handling a client's request is shown in alg. 1 at the top of the page.

According to this algorithm, lookup requests are processed the same way as in the centralized, single-node name service. Only updates are passed around the ring (unless they are impossible to execute according to the local servers tables - in that case it is pointless to send them to the other ring members). Thus, all servers update their tables before the client receives the update acknowledgement.

In order to define an algorithm for processing the incoming requests by the server-to-server channel, we first need to take a closer look at the communication between name servers. Let's suppose a name server NS_i wants to send the request around the ring. It contacts the next available server in the ring NS_{i+1} and sends the request. NS_{i+1} forwards the request to NS_{i+2}, which sends it to NS_{i+3} and so on; finally the request gets back to NS_i.

Initially, NS_i sends the request to NS_{i+1}. NS_{i+1} sends back a 'got ack', which tells NS_i, that NS_{i+1} received the request. At the same time NS_{i+1} forwards the request to NS_{i+2} and waits for it's 'got ack' message. When the 'got ack' from NS_{i+2} is received by NS_{i+1}, NS_{i+1} sends 'fwd ack' to let NS_i know it's request has been successfully forwarded[2]. The algorithm for handling a request coming from server-to-server channel is shown in alg. 2.

[2] In general, when the ring consists of n nodes and we consider sending one message as a time unit, it takes n time units for a request to traverse the ring.

Algorithm 2 Process a server's request.

1. Read the request.
2. If the request originated from this host, send 'got ack' and 'forward ack', perform appropriate actions on the client side and STOP.
3. If someone else is the sender, do the following:
4. Send 'got ack' and send the request to the next server in the ring.
5. Wait for 'got ack' from the next server, and update the local tables according to the request.
6. When it comes, send 'fwd ack' to the previous server.
7. Wait for 'fwd ack' from the next one.
8. When it arrives, consider the request to be forwarded successfully and STOP.

Algorithm 3 Register a DVM (simplified version).

1. Read the request from the client.
2. Check if there is a server for this DVM registered in the local tables.
3. If there is, execute steps 4-10, otherwise goto 11.
4. Check if the server host and port are equal to those from the request.
5. If they are, consider this request to be a refresh of the entry, so reset the timeout and send the request around the ring. When it comes from the other side, send the acknowledgement to the client and STOP.
6. If the answer in 4 is 'no', do the following:
7. Try to ping the current DVM server.
8. If the server is alive, send the information about it to the client and STOP.
9. If the ping failed, send a DPING request (see below) around the ring. Wait until it comes from the other side. If the DPING succeeded, the current DVM server is alive - send appropriate information to the client and STOP.
10. If PING and DPING failed, register the client as a new DVM server, send the request around the ring and when it comes from the other side, send the acknowledgement to the client. Then STOP.
11. Register the client as a new DVM server, send the request around the ring and when it comes back send the acknowledgement to the client. Then STOP.

This is a very general form of the algorithm, details depend on the request type. Some additional actions have to be taken when a server does not receive 'got ack' or 'fwd ack' from the next server in a specified amount of time or when it's not possible to contact the neighbor, those mechanisms are described in section 4.

Step 2 requires further explanation. If a server receives a request and detects it is the sender of the request, then this means that the request traversed the entire ring and was received by all name servers. Therefore it shouldn't be forwarded to the next ring member, because it would cause the message to go around the ring once more. The sender of the message doesn't forward it, but the previous node in the ring still waits for the acknowledgements. If it doesn't receive them, it will assume the server crashed. To avoid that, the sender of the message sends back 'got ack' and 'fwd ack' despite the fact that it did not forward the message.

We now formulate the algorithms for handling different types of client requests. For a request to register a DVM server, the algorithm is shown in alg. 3.

This is a simplified version since under some circumstances there must be some more actions taken. Those special cases are described in section 4. The algorithm for registering a dormant daemon is much simpler and is shown in alg. 4.

Algorithm 4 Register a dormant daemon.

1. Read the request from the client.
2. Put the data into the local tables and reset the timeout.
3. Send the request around the ring. When it comes back, send the acknowledgement to the client and STOP.

As noted before, a list of hosts running dormant daemons is maintained by the name service. Thus, it is unnecessary to check if a particular host has been already registered. If it was, then the entry will be overwritten with the same value.

Recall that lookups do not require an interaction between name servers. This is because all updates are immediately propagated around the ring, therefore all name servers keep exactly the same data and no communication between them is necessary to give an answer to the client.

It is noteworthy at this point to mention the DPING operation, which is specific for the distributed name service and does not appear in the centralized one. When a DPING is requested by one of the ring members, this means that it hasn't been able to contact a DVM server, but that does not necessarily means the DVM server has crashed. The name server asks all the other ring members to ping the DVM server on its behalf. Thus, the algorithm for processing the DPING request is shown in alg. 5.

When the DPING request traverses the ring and finally returns to it's sender, the sender checks the value of the 'pinged' field. If the value is 'true', it means at least one name server was able to contact the DVM server, so the server is alive. If 'pinged' contains 'false', the DVM server is considered to have crashed or is in a minority partition.

Note, that after the first ring member successfully pinged the DVM server, the subsequent name servers do not attempt to contact it again, simply, the request is forwarded. An example of a DPING operation is shown in fig. 2 below.

In this example, NS2 was unable to ping DVMS3, so it requested a DPING. NS3 wasn't able to ping DVMS3 either, so it forwarded the request with 'pinged' set to 'false'. Finally NS4 managed to ping DVMS3, so it set 'pinged' to 'true' in the request. All subsequent name servers forwarded the request, which finally reached NS2.

4 Fault Tolerance In HDNS

The Harness Distributed Name Service is not fully fault-tolerant (although it is much more reliable than a single-node name service). Unfortunately the distribution itself introduces additional types of faults. In the centralized name service, a machine crash

Algorithm 5 DPING.

1. Read the DPING request from server-to-server channel.
2. If the 'pinged' field in the request is set to 'true', then that means one of the previous name servers has managed to contact the DVM server. In this case forward the request without additional actions and STOP.
3. Otherwise, attempt to ping the DVM server.
4. If the ping succeeded, then set the 'pinged' field in the request to 'true' and forward the request. Then STOP.
5. If the ping failed, leave 'false' in the 'pinged' field, forward the request and STOP.

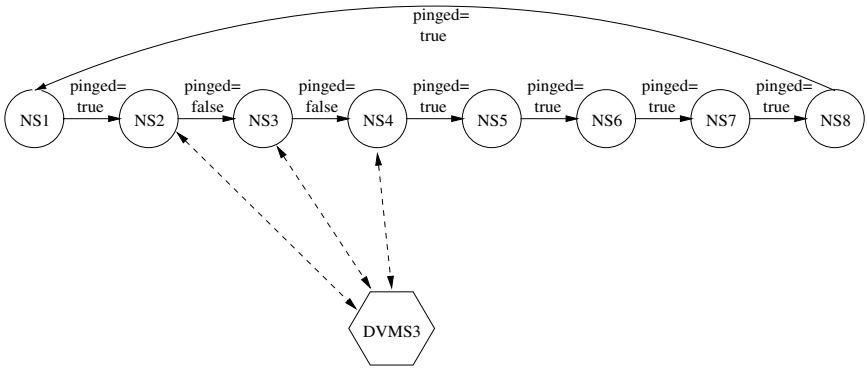


Fig. 2. An Example Of DPING Operation.

or a link failure was the only problems that could cause the whole system to crash. In the distributed version, data can become inconsistent which may be even more dangerous than a node crash. In this section we consider possible scenarios and discuss how HDNS deals with them.

4.1 Message Loss

The conditions in which a request traversing the ring may be lost can be described as: i) if there were no acknowledgements or ii) a single machine crash or a link failure would cause the request to traverse the ring.

Consider that P_{mf} is the probability that a host in the ring crashes exactly at the moment when the message reaches it and before it's able to forward it. P_{lf} will denote the probability that a link between two nodes does not operate. When there are no acknowledgements, the probability that a message is lost equals $P_l = P_{mt} + \frac{1}{n-1}P_{lt}^2$ i.e. if a ring member crashes or becomes isolated when the message reaches it, the message will never make around the ring.

In the version with acknowledgements (as described above), one of the following may take place to cause the message loss:

Recall, that if a name server sending or forwarding a request doesn't receive a 'got ack' or 'fwd ack' message, it assumes that the recipient has crashed and connects to the next server in the ring, circumventing the failed host. To cause message loss, we must visualize a situation in which all acknowledgements have been sent but the message has not been forwarded.

In scenario 1, NS_{i+1} receives the request from NS_i, sends the 'got ack' and forwards the request to NS_{i+2}. After getting 'got ack' from NS_{i+2}, it sends 'fwd ack' to NS_i. From that moment NS_i is convinced the message has been successfully forwarded. Suppose NS_{i+2} sent the 'got ack' but it took an arbitrarily long time to contact the next ring member, so that NS_{i+1} had enough time to send 'fwd ack' while the message is still in NS_{i+2}. Just after NS_{i+1} had sent 'fwd ack', both NS_{i+1} and NS_{i+2} crash (before NS_{i+2} forwarded the message). The only host who could detect the fact that NS_{i+2} hasn't forwarded the message is NS_{i+1}, because it will not get 'fwd ack' from NS_{i+2}. But NS_{i+1} has crashed too, and NS_i got both acknowledgements and will not take care

about the message any longer. In this situation the message is lost and will return to the sender. This situation may occur with the following probability:

$$P_{ldh} = \frac{1}{n-1} P_{mf}^2$$

where P_{mf} is the probability that a single ring member crashes exactly at the moment described in the scenario above.

In scenario 2, the situation is similar, but instead of a host crashing, a link fails. Suppose the connection between the pair NS_{i+1}, NS_{i+2} and all the other ring members stopped operating as soon as NS_{i+1} sent 'fwd ack' to NS_i, while the message is still in NS_{i+2}. In that case the message is 'trapped' in the pair NS_{i+1}, NS_{i+2} and will not traverse further (in fact it will be discarded as soon as NS_{i+2} detects that there is no connection to the message sender and considers the sender to have crashed. The probability of the occurrence of such a situation is:

$$P_{ldl} = \frac{1}{n-1} P_{lf}^2$$

So the model with acknowledgements is as vulnerable to faults, as the model without acknowledgements, but it is at less risk for node crashes. Therefore the probability that a message is lost in HDNS is:

$$P_{ld} = P_{ldh} + P_{ldl} = \frac{1}{n-1}(P_{mf}^2 + P_{lf}^2) < P_l$$

The probability of message loss is much less in the model with acknowledgements, than in the model without them, and notice that the total time necessary for the request to traverse the ring is equal in both cases. Evidently, it requires more messages to be sent, but the request gets back to the sender in the same amount of time in both models.

In case a message has been lost, a thread is created on the sender node for each request sent. If the request doesn't come back in a specified amount of time, the thread re-sends it. So, even if a message is lost as a consequence of two name servers crashing, it will be retransmitted. Thus, to the client it will appear as an increase in processing time.

4.2 Data Inconsistency

As noted above, all the update requests have to be processed in the same manner regardless of which ring members have been contacted by a client. They should also produce consistent answers when they receive lookup requests. The multiple injection points may be an issue when we consider registering DVM servers. Assume two DVM servers attempt to register themselves as servers for the same DVM (let's call it 'Red'). As it was said in section 2, each Harness DVM consists of exactly one DVM server and an arbitrary number of kernels. If both servers attempt to register themselves using the same ring node, there is no problem - the first is registered, and the second one receives information that there is a DVM server for 'Red' already. Suppose then, that they use different nodes to register themselves and there is no DVM server for 'Red' registered, so theoretically both name servers should accept the request according to the contents of their tables. But according to alg. 3, the DVM server being registered gets the acknowledgement from the name server after the request comes back to the name server, and that creates an opportunity to introduce a level of control, thus solving the problem.

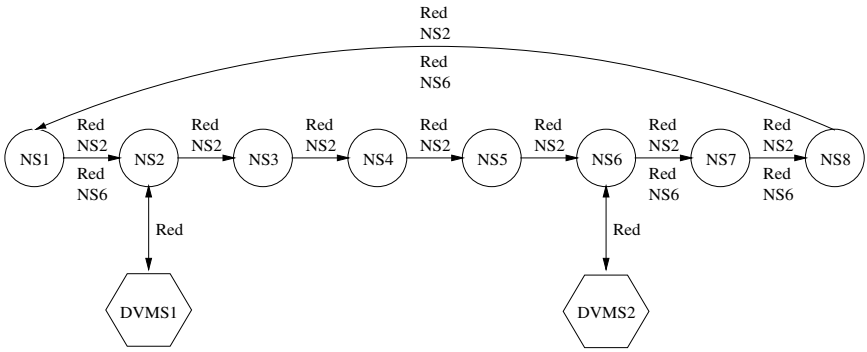


Fig. 3. Priority Based DVM Registration.

As an example, consider the following scenario. NS2 sends it's request around the ring and NS6 will do the same. Both requests contain the same DVM name. It must happen, that the request of NS6 reaches NS2 and the request of NS2 reaches NS6 (not necessarily at the same time, but its certain that the request of NS6 will reach NS2 before it comes back to NS6 and the request of NS2 will reach NS6 before it comes back to NS2). At that moment the server priorities become significant. First consider NS2. It receives the request of NS6 and detects that there is its own request pending for the same DVM. So it compares it's own priority with the priority of the received requests sender (NS6). If it's own priority is higher (it is in our example), it discards the incoming message; i.e. it is not forwarded by NS2 and will never return to NS6. This does not mean that NS6 will never return an answer to DVMS2. However, the request of NS6 will never return to it, NS6 receives the request of NS2 and performs exactly the same procedure. It detects that there is it's own request pending for 'Red' and compares the priorities. The NS6 priority is less than NS2, so NS6 modifies the entry for 'Red' in it's table, forwards the message and instructs DVMS2 that there is another server for 'Red'. This is shown in fig. 3. The request sent by NS2 is above the arrows in this diagram, whilst the request of NS6 is shown below the arrows.

One potential problem in this algorithm is that for a short time, name servers NS7, NS8 and NS1 have invalid entries for 'Red' in their tables and, if asked, will return false answers. This is the time after those servers received the request of NS6 and before they received the request of NS2. To partially avoid this situation, the clients maintain a sorted array of name servers and they first try to contact the server with the highest priority. This does not guarantee that none of the servers with false data will be asked, but it decreases the probability that the situation will arise.

5 Conclusions

A significant effort went into deciding which kind of topology should be used to build the HDNS. All of them have disadvantages and we finally decided to implement the service as a ring of servers with multiple injection points. The working prototype proves the concept. The current Harness release (v 1.7) works with the new name service without any changes in implementation (it just uses one of the ring members). In the next

release (2.0), Harness core classes will be changed so that they will take advantage of the HDNS.

During the testing phase of the prototype there appeared several of the problems described above. After some experimentation it turned out, that in most cases the DNS lookup was causing delays in communication. The host name lookup table solved this problem, while an 'interrupting' thread corrected the issue with the long time of opening a connection when a host is unreachable. After changing the process of opening a connection as described above, the problem no longer occurs.

The new name service, as well as the old one and most of the other Harness services, use object input / output streams to send data. In other words, data being sent has the form of serialized Java objects. The process of serialization and de-serialization in Java takes some time, but fortunately name service do not send large amounts of data, so the serialization does not affect the communication speed.

References

1. M. Migliardi and V. Sunderam. Heterogenous Distributed Virtual Machines In The Harness Metacomputing Framework. In *Proc. of the Eigth Heterogeneous Computing Workshop*, April 1999.
2. M. Migliardi and V. Sunderam. The Harness Metacomputing Framework. In *Proc. of the Ninth SIAM Conference on Parallel Processing for Scientific Computing*, March 1999.

Fault Tolerant MPI for the HARNESS Meta-computing System

Graham E. Fagg, Antonin Bukovsky,and Jack J. Dongarra

Department of Computer Science, Suite 203, 1122 Volunteer Blvd.,
University of Tennessee, Knoxville, TN-37996-3450, USA.

fagg@cs.utk.edu

Abstract. Initial versions of MPI were designed to work efficiently on multiprocessors which had very little job control and thus static process models. Subsequently forcing them to support a dynamic process model suitable for use on clusters or distributed systems would have reduced their performance. As current HPC collaborative applications increase in size and distribution the potential levels of node and network failures increase the need arises for new fault tolerant systems to be developed. Here we present a new implementation of MPI called FT-MPI that allows the semantics and associated modes of failures to be explicitly controlled by an application via a modified MPI API. Given is an overview of the FT-MPI semantics, design, example applications and some performance issues such as efficient group communications and complex data handling.

1 Introduction

Although MPI [11] is currently the de-facto standard system used to build high performance applications for both clusters and dedicated MPP systems, it is not without it problems. Initially MPI was designed to allow for very high efficiency and thus performance on a number of early 1990s MPPs, that at the time had limited OS runtime support. This led to the current MPI design of a static process model. While this model was possible to implement for MPP vendors, easy to program for, and more importantly something that could be agreed upon by a standards committee.

The MPI static process model suffices for small numbers of distributed nodes within the currently emerging masses of clusters and several hundred nodes of dedicated MPPs. Beyond these sizes the mean time between failure (MTBF) of CPU nodes starts becoming a factor. As attempts to build the next generation Peta-flop systems advance, this situation will only become more adverse as individual node reliability becomes out weighted by orders of magnitude increase in node numbers and hence node failures.

The aim of FT-MPI is to build a fault tolerant MPI implementation that can survive failures, while offering the application developer a range of recovery options other than just returning to some previous check-pointed state. FT-MPI is built on the

V.N. Alexandrov et al. (Eds.): ICCS 2001, LNCS 2073, pp. 355-366, 2001.
© Springer-Verlag Berlin Heidelberg 2001

HARNESS [1] meta-computing system, and is meant to be used as its default application level message passing interface.

2 Check-Point and Roll Back versus Replication Techniques

The first method attempted to make MPI applications fault tolerant was through the use of check-pointing and roll back. Co-Check MPI [2] from the Technical University of Munich being the first MPI implementation built that used the Condor library for check-pointing an entire MPI application. In this implementation, all processes would flush their messages queues to avoid in flight messages getting lost, and then they would all synchronously check-point. At some later stage if either an error occurred or a task was forced to migrate to assist load balancing, the entire MPI application would be rolled back to the last complete check-point and be restarted. This systems main drawback being the need for the entire application having to check-point synchronously, which depending on the application and its size could become expensive in terms of time (with potential scaling problems). A secondary consideration was that they had to implement a new version of MPI known as tuMPI as retro-fitting MPICH was considered too difficult.

Another system that also uses check-pointing but at a much lower level is StarFish MPI [3]. Unlike Co-Check MPI which relies on Condor, Starfish MPI uses its own distributed system to provide built in check-pointing. The main difference with Co-Check MPI is how it handles communication and state changes which are managed by StarFish using strict atomic group communication protocols built upon the Ensemble system [4], and thus avoids the message flush protocol of Co-Check. Being a more recent project StarFish supports faster networking interfaces than tuMPI.

The project closest to FT-MPI known by the author is the Implicit Fault Tolerance MPI project MPI-FT [15] by Paraskevas Evripidou of Cyprus University. This project supports several master-slave models where all communicators are built from grids that contain ‘spare’ processes. These spare processes are utilized when there is a failure. To avoid loss of message data between the master and slaves, all messages are copied to an observer process, which can reproduce lost messages in the event of any failures. This system appears only to support SPMD style computation and has a high overhead for every message and considerable memory needs for the observer process for long running applications.

3 FT-MPI Semantics

Current semantics of MPI indicate that a failure of a MPI process or communication causes all communicators associated with them to become *invalid*. As the standard provides no method to reinstate them (and it is unclear if we can even *free* them), we are left with the problem that this causes MPI_COMM_WORLD itself to become invalid and thus the entire MPI application will grid to a halt.

FT-MPI extends the MPI communicator states from {valid, invalid} to a range {FT_OK, FT_DETECTED, FT_RECOVER, FT_RECOVERED, FT_FAILED}. In essence this becomes {OK, PROBLEM, FAILED}, with the other states mainly of interest to the internal fault recovery algorithm of FT_MPI. Processes also have typical states of {OK, FAILED} which FT-MPI replaces with {OK, Unavailable, Joining, Failed}. The *Unavailable* state includes unknown, unreachable or "we have not voted to remove it yet" states.

A communicator changes its state when either an MPI process changes its state, or a communication within that communicator fails for some reason. Some more detail on failure detection is given in 4.4.

The typical MPI semantics is from OK to Failed which then causes an application abort. By allowing the communicator to be in an intermediate state we allow the application the ability to decide how to alter the communicator and its state as well as how communication within the intermediate state behaves.

3.1 Failure Modes

On detecting a failure within a communicator, that communicator is marked as having a probable error. Immediately as this occurs the underlying system sends a state update to all other processes involved in that communicator. If the error was a communication error, not all communicators are forced to be updated, if it was a process exit then all communicators that include this process are changed. Note, this might not be all current communicators as we support MPI-2 dynamic tasks and thus multiple MPI_COMM_WORLDS.

How the system behaves depends on the communicator failure mode chosen by the application. The mode has two parts, one for the communication behavior and one for the how the communicator reforms if at all.

3.2 Communicator and Communication Handling

Once a communicator has an error state it can only recover by rebuilding it, using a modified version of one of the MPI communicator build functions such as MPI_Comm_{create, split or dup}. Under these functions the new communicator will follow the following semantics depending on its failure mode:

SHRINK: The communicator is reduced so that the data structure is contiguous. The ranks of the processes are **changed**, forcing the application to recall MPI_COMM_RANK.

- BLANK: This is the same as SHRINK, except that the communicator can now contain gaps to be filled in later. Communicating with a gap will cause an invalid rank error. Note also that calling MPI_COMM_SIZE will return the extent of the communicator, not the number of valid processes within it.
- REBUILD: Most complex mode that forces the creation of new processes to fill any gaps until the size is the same as the extent. The new processes can either be places in to the empty ranks, or the communicator can be shrank and the remain-

ing processes filled at the end. This is used for applications that require a certain size to execute as in power of two FFT solvers.
- ABORT: Is a mode which affects the application immediately an error is detected and forces a graceful abort. The user is unable to trap this. If the application need to avoid this they must set all communicators to one of the above communicator modes.

Communications within the communicator are controlled by a message mode for the communicator which can be either of:
- NOP: No operations on error. I.e. no user level message operations are allowed and all simply return an error code. This is used to allow an application to return from any point in the code to a state where it can take appropriate action as soon as possible.
- CONT: All communication that is NOT to the affected/failed node can continue as normal. Attempts to communicate with a failed node will return errors until the communicator state is reset.

The user discovers any errors from the return code of any MPI call, with a new fault indicated by MPI_ERR_OTHER. Details as to the nature and specifics of an error is available though the cached attributes interface in MPI.

3.3 Point to Point versus Collective Correctness

Although collective operations pertain to point to point operations in most cases, extra care has been taken in implementing the collective operations so that if an error occurs during an operation, the result of the operation will still be the same as if there had been no error, or else the operation is aborted.

Broadcast, gather and all gather demonstrate this perfectly. In Broadcast even if there is a failure of a receiving node, the receiving nodes still receive the same data, i.e. the same end result for the surviving nodes. Gather and all-gather are different in that the result depends on if the problematic nodes sent data to the gatherer/root or not. In the case of gather, the root might or might not have gaps in the result. For all gather which typically uses a ring algorithm it is possible that some nodes may have complete information and others incomplete. Thus for operations that require multiple node input as in gather/reduce type operations any failure causes all nodes to return an error code, rather than possibly invalid data. Currently an addition flag controls how strict the above rule is enforced by utilizing an extra barrier call at the end of the collective call if required.

3.4 FT-MPI Usage

Typical usage of FT-MPI would be in the form of an error check and then some corrective action such as a communicator rebuild. A typical code fragment is shown below, where on an error the communicator is simply rebuilt and reused:

```
  rc= MPI_Send (----, com);
  If (rc==MPI_ERR_OTHER)
      MPI_Comm_dup (com, newcom);
      com = newcom;     /* continue.. */
```

Some types of computation such as SPMD master-slave codes only need the error checking in the master code if the user is willing to accept the master as the only point of failure. The example below shows how complex a master code can become. In this example the communicator mode is BLANK and communications mode is CONT. The master keeps track of work allocated, and on an error just reallocates the work to any 'free' surviving processes. Note, the code checks to see if there are surviving worker processes left after each death is detected.

```
  rc = MPI_Bcast ( initial_work…);
  if(rc==MPI_ERR_OTHER)reclaim_lost_work(…);

  while ( ! all_work_done) {
    if (work_allocated) {
      rc = MPI_Recv ( buf, ans_size, result_dt,
                      MPI_ANY_SOURCE,         MPI_ANY_TAG,
comm, &status);
      if (rc==MPI_SUCCESS) {
                        handle_work (buf);
                        free_worker (status.MPI_SOURCE);
                        all_work_done--;
                        }
      else {
           reclaim_lost_work(status.MPI_SOURCE);
            if (no_surviving_workers) { /* ! do something
! */ }
           }

    } /* work allocated */

  /* Get a new worker as we must have received a result
or a death */
    rank=get_free_worker_and_allocate_work();
    if (rank) {
      rc = MPI_Send (… rank… );
      if (rc==MPI_OTHER_ERR) reclaim_lost_work (rank);
      if (no_surviving_workers) { /* ! do something ! */
}
    } /* if free worker */

  } /* while work to do */
```

4 FT_MPI Implementation Details

FT-MPI is a partial MPI-2 implementation in its own right. It currently contains support for both C and Fortran interfaces, all the MPI-1 function calls required to run both the PSTSWM [6] and BLACS applications. BLACS is supported so that SCALAPACK application can be tested. Currently only some the dynamic process control functions from MPI-2 are supported.

The current implementation is built as a number of layers as shown in figure 1. Operating system support is provided by either PVM or the C HARNESS G_HCORE. Although point to point communication is provided by a modified SNIPE_Lite communication library taken from the SNIPE project [4].

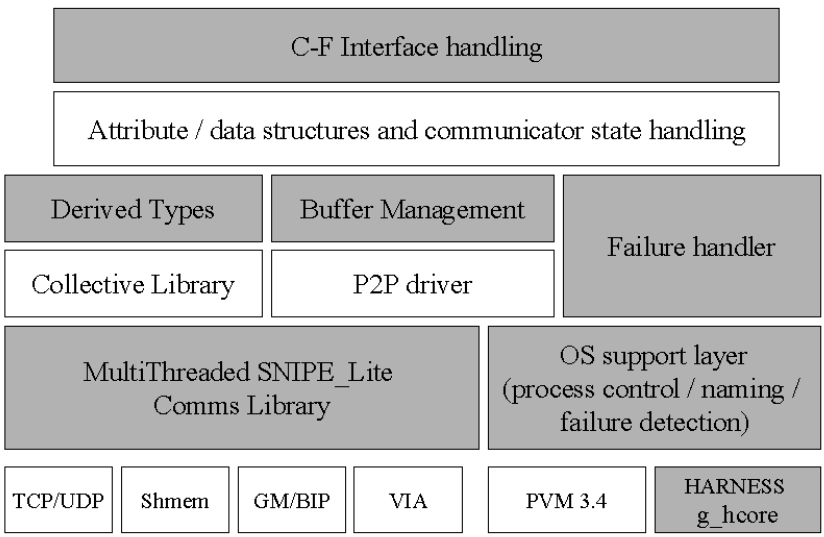


Fig 1. Overall structure of the FT-MPI implementation

A number of components have been extensively optimized, these include:

- Derived data types and message buffers.
- Collective communications.
- Point to point communication using multi-threading.

4.1 Derived Data Type Handling

MPI-1 introduced extensive facilities for user Derived DataType (DDT)[11] handling that allows for strongly typed message passing. The handling of these possibly non-contiguous data types is very important in real applications, and is often a neglected area of communication library design [17]. Most communications libraries are de-

signed for low latency and/or high bandwidth with contiguous blocks of data [14]. Although this means that they must avoid unnecessary memory copies, the efficient handling of recursive data structures is often left to simple iterations of a loop that packs a send/receive buffer.

FT-MPI DDT handling. Having gained experience with handling DDTs within a heterogeneous system from the PVMPI/MPI_Connect library [18] the authors of FT-MPI redesigned the handling of DDTs so that they would not just handle the recursive data-types flexibly but also take advantage of internal buffer management structure to gain better performance. In a typical system the DDT would be collected/gathered into a single buffer and then passed to the communications library, which may have to encode the data using XDR for example, and then segment the message into packets for transmission. These steps involving multiple memory copies across program modules (reducing cache effectiveness) and possibly precluding overlapping (concurrency) of operations.

The DDT system used by FT-MPI was designed to reduce memory copies while allowing for overlapping in the three stages of data handling:

- gather/scatter : Data is collected into or from recursively structured non-contiguous memory.
- encoding/decoding : Data passed between heterogeneous machine architectures than use different floating point representations need to be converted so that the data maintains the original meaning.
- send/receive packetizing : All of the send or receive cannot be completed in a single attempt and the data has to be sent in blocks. This is usually due to buffering constraints in the communications library/OS or even hardware flow control.

DDT methods and algorithms. Under FT-MPI data can be gathered/scattered by compressing the data type representation into a compacted format that can be efficiently transversed (not to be confused with compressing data discussed below). The algorithm used to compact data type representation would break down any recursive data type into an optimized maximum length new representation. FT-MPI checks for this optimization when the users application commits the data type using the MPI_Type_commit API call. This allows FT-MPI to optimize the data type representation before any communication is attempted that uses them.

When the DDT is being processed the actual user data itself can also be compacted into/from a contiguous buffer. Several options for this type of buffering are allowed that include:

- Zero padding: Compacting into the smallest buffer space
- Minimal padding: Compacting into smallest space but maintaining correct word alignment
- Re-ordering pack: Re-arranging the data so that all the integers are packed first, followed by floats etc. i.e. type by type.

The minimal and no padded methods are used when moving the data type within a homogeneous set of machines that require no numeric representation encoding or

decoding. The zero padding method benefits slower networks, and alignment padded can in some cases assist memory copy operations, although its real benefit is when used with re-ordering.

The re-ordered compacting method shown in figure 2, is designed to be used when some additional form encoding/decoding takes place. In particular moving the re-ordered data, type by type through fixed XDR buffers improves its performance considerably. Two types of DDT encoding are supported, the first is the slower generic SUN XDR format and the second is simple byte swapping to convert between little and big endian numbers.

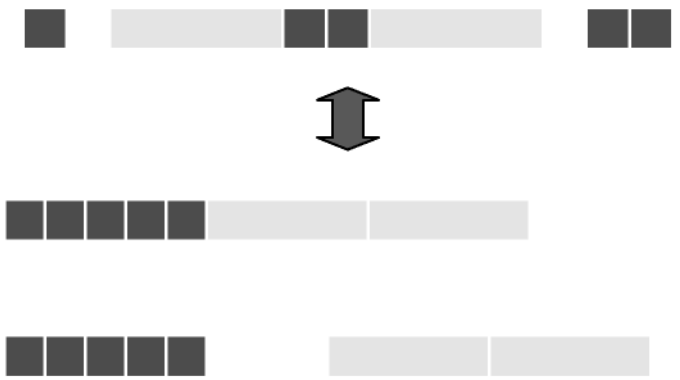


Fig. 2. Compacting storage of re-ordered DDT. Without padding, and with correct alignment

FT-MPI DDT performance. Tests comparing the DDT code to MPICH (1.3.1) on a ninety three element DDT taken from a fluid dynamic code were performed between Sun SPARC Solaris and Red Hat (6.1) Linux machines as shown in table 1 below. The tests were on small and medium arrays of this data type. All the tests were performed using MPICH MPI_Send and MPI_Recv operations, so that the point to point communications speeds were not a factor, and only the handling of the data types was compared.

The tests show that the compacted data type handling gives from 10 to 19% improvement for small messages and 78 to 81% for larger arrays on same numeric representation machines. The benefits of buffer reuse and re-ordered data elements leads to considerable improvements on heterogeneous networks however. Noting that this test used MPICH to perform the point to point communication, and thus the overlapping of the data gather/scatter, encoding/decoding and non-blocking communication is not shown here, and is expected to yield even higher performance.

Table 1. Performance of the FT-MPI DDT software compared to MPICH.

Type of operation (arch 2 arch) (method) (encoding)	11956 bytes B/W MB/Sec	%compared to MPICH	95648 bytes B/W MB/Sec	%compared to MPICH
Sparc 2 Sparc MPICH	5.49		5.47	
Sparc 2 Sparc DDT	6.54	+19 %	9.74	+78 %
Linux 2 Linux MPICH	7.11		8.79	
Linux 2 Linux DDT	7.87	+10 %	9.92	+81 %
Sparc 2 Linux MPICH	0.855		0.729	
Sparc 2 Linux DDT Byte Swap	5.87	+586 %	8.20	+1024 %
Sparc 2 Linux DDT XDR	5.31	+621 %	6.15	+ 743 %

FT-MPI DDT additional benefits and future. The above tests were performed using the DDT software as a standalone library that can be used to improve any MPI implementation. This software is being made into a true MPI profiling library so that its use will be completely transparent. Two other efforts closely parallel this section of work on DDTs. PACX [19] from HLRS, RUS Stuttgart, requires the heterogeneous data conversion facilities and a project from NEC Europe [16] concentrates on efficient data type representation and transmission in homogeneous systems.

4.2 Collective Communications

The performance of the MPI's collective communications is critical to most MPI-based applications [6]. A general algorithm for a given collective communication operation may not give good performance on all systems due to the differences in architectures, network parameters and the storage capacity of the underlying MPI implementation [7]. In an attempt to improve over the usual collective library built on point to point communications design as in the logP model [9], we built a collective communications library that is tuned to its target architecture though the use a limited set of micro benchmarks. Once the static system is optimized we then tune the topology dynamically by re-orders the logical addresses to compensate for changing run time variations. Other projects that use a similar approach to optimizing include [12] and [13]. Further details and performance results for our method can be found in [8]and [10].

4.3 Point to Point Multi-threaded Communications

FT-MPIs requirements for communications have forced us to use a multi-threaded communications library. The three most important criteria were:

- High performance networking is not affected by concurrent use of slower networking (Myrinet versus Ethernet)
- Non-blocking calls make progress outside of API calls
- Busy wait (CPU spinning) is avoided within the runtime library

To meet these requirements, in general communication requests are passed to a thread via a shared queue to be completed unless the calling thread can complete the operation immediately. Receives are placed into a pending queue by a separate thread. There is one sending and receiving thread per type of communication media. I.e. a thread for TCP communications, a thread for VIA and a thread for handling GM message events. The collective communications are built upon this point to point library.

4.4 Failure Detection

It is important to note that the failure handler shown in figure 1, gets notification of failures from both the point to point communications libraries as well as the OS support layer from the HARNESS G_HCORE. Communication errors are usually detected when a communication with a failed party is flagged before the failed parties OS layer has managed to propagated the failure signal via any low level services. The handler is responsible for notifying all tasks of errors as they occur by injecting notify messages into the send message queues ahead of user level messages.

4.5 OS Support and the HARNESS G_HCORE

When FT-MPI was first designed the only HARNESS Kernel available was an experiment Java implementation from Emory University [5]. Tests were conducted to implement required services on this from C in the form of C-Java wrappers that made RMI calls. Although they worked, they were not very efficient and so FT-MPI was instead initially developed using the readily available PVM system.

As the project has progressed, the primary author developed the G_HCORE, a C based HARNESS core library that uses the same policies as the Java version. This core allows for services to be built that FT-MPI requires.

Current services used by FT-MPI break down into four categories:

- Spawn and Notify service. This plug-in allows remote processes to be initiated and then monitored. The service notifies other interested processes when a failure or exit of the invoked process occurs.
- Naming services. These allocate unique identifiers in a distributed environment.
- Distributed Replicated Database (DRD). This service allows for system state and additional MetaData to be distributed, with replication specified at the record level.

5 Conclusions

FT-MPI is an attempt to provide application programmers with different methods of dealing with failures within MPI application than just check-point and restart. It is hoped that by experimenting with FT-MPI, new applications methodologies and algorithms will be developed to allow for both high performance and the survivability required by the next generation of terra-flop and beyond machines.

FT-MPI in itself is already proving to be a useful vehicle for experimenting with self-tuning collective communications, distributed control algorithms, various dynamic library download methods and improved sparse data handling subsystems, as well as being the default MPI implementation for the HARNESS project.

References

1. Beck, Dongarra, Fagg, Geist, Gray, Kohl, Migliardi, K. Moore, T. Moore, P. Papadopoulous, S. Scott, V. Sunderam, "HARNESS: a next generation distributed virtual machine", Journal of Future Generation Computer Systems, (15), Elsevier Science B.V., 1999.
2. G. Stellner, “CoCheck: Checkpointing and Process Migration for MPI”, In Proceedings of the International Parallel Processing Symposium, pp 526-531, Honolulu, April 1996.
3. Adnan Agbaria and Roy Friedman, “Starfish: Fault-Tolerant Dynamic MPI Programs on Clusters of Workstations”, In the 8th IEEE International Symposium on High Performance Distributed Computing, 1999.
4. Graham E. Fagg, Keith Moore, Jack J. Dongarra, "Scalable networked information processing environment (SNIPE)", Journal of Future Generation Computer Systems, (15), pp. 571-582, Elsevier Science B.V., 1999.
5. Mauro Migliardi and Vaidy Sunderam, “PVM Emulation in the HARNESS MetaComputing System: A Plug-in Based Approach”, Lecture Notes in Computer Science (1697), pp 117-124, September 1999.
6. P. H. Worley, I. T. Foster, and B. Toonen, “Algorithm comparison and benchmarking using a parallel spectral transform shallow water model”, Proccedings of the Sixth Workshop on Parallel Processing in Meteorology, eds. G.-R. Hoffmann and N. Kreitz, World Scientific, Singapore, pp. 277-289, 1995.
7. Thilo Kielmann, Henri E. Bal and Segei Gorlatch. Bandwidth-efficient Collective Communication for Clustered Wide Area Systems. *IPDPS 2000*, Cancun, Mexico. (May 1-5, 2000)
8. Graham E. Fagg, Sathish S. Vadhiyar, Jack J. Dongarra, “ACCT: Automatic Collective Communication Tuning”, Proc of the 7th European PVM/MPI Users’ Group Meeting, Lecture Notes in Computer Science, Vol. 1908, Springer Verlag, pp. 354-361, September 2000.
9. David Culler, R. Karp, D. Patterson, A. Sahay, K.E. Schauser, E. Santos, R. Subramonian and T. von Eicken. LogP: Towards a Realistic Model of Parallel Computation. In Proc. Symposium on Principles and Practice of Parallel Programming (PpoPP), pages 1-12, San Diego, CA (May 1993).
10. Sathish S. Vadhiyar, Graham E. Fagg and Jack J. Dongarra, “Automatically Tuned Collective Communications”, *Proc. of SuperComputing 2000*, Dallas, Texas, November 2000.

11. Marc Snir, Steve Otto, Steven Huss-Lederman, David Walker and Jack Dongarra. MPI-The Complete Reference. Volume 1, The MPI Core, second edition (1998).
12. M. Frigo. FFTW: An Adaptive Software Architecture for the FFT. Proceedings of the ICASSP Conference, page 1381, Vol. 3. (1998).
13. R. Clint Whaley and Jack Dongarra. Automatically Tuned Linear Algebra Software. SC98: High Performance Networking and Computing. http://www.cs.utk.edu/~rwhaley/ATL/INDEX.HTM. (1998)
14. L. Prylli and B. Tourancheau. "BIP: a new protocol designed for high performance networking on myrinet" In the PC-NOW workshop, IPPS/SPDP 1998, Orlando, USA, 1998.
15. Soulla Louca, Neophytos Neophytou, Adrianos Lachanas, Paraskevas Evripidou, "MPI-FT: A portable fault tolerance scheme for MPI", Proc. of PDPTA '98 International Conference, Las Vegas, Nevada 1998.
16. Jesper Lasson Traff, Rolf Hempel, Hubert Ritzdort and Falk Zimmermann, "Flattening on the Fly: Efficient Handling of MPI Derived Datatypes", Proc of the 6th European PVM/MPI Users' Group Meeting, Lecture Notes in Computer Science, Vol. 1697, Springer Verlag, pp. 109-116, Bareclona, September 1999.
17. W.D. Gropp, E. Lusk and D. Swider, "Improving the performance of MPI derived datatypes", In Third MPI Developer's and User's Conf (MPIDC'99), pp. 25-30, 1999.
18. Graham E Fagg, Kevin S. London and Jack J. Dongarra, "MPI_Connect, Managing Heterogeneous MPI Application Interoperation and Process Control", EuroPVM-MPI 98, Lecture Notes in Computer Science, Vol. 1497, pp.93-96, Springer Verlag, 1998.
19. Edgar Gabriel, Michael Resch, Thomas Beisel and Rainer Keller, "Distributed Computing in a Heterogeneous Computing Environment", EuroPVM-MPI 98, Lecture Notes in Computer Science, Vol. 1497, pp.180-187, Springer Verlag, 1998.

A Harness Control Application for Hand-Held Devices

Tomasz Tyrakowski†, Vaidy Sunderam† and Mauro Migliardi†

†Department of Math & Computer Science
Emory University
Atlanta, GA
30302
{ttomek | vss | om}@mathcs.emory.edu

Abstract. This document describes the design of a Harness control application for hand-held devices. Some features and future directions are covered, as well as the limitation caused by the current state of the development tools for mobile computers. Section 1 describes the general idea of mobile interfaces to the Harness metacomputing system. In section 2 some of the mobile control application design issues for Harness are described, while section 3 analyzes the functionality of this application. Section 4 covers some other possibilities of the users collaboration utilizing Harness proxy plugins. Section 5 gives our conclusions about the most appropriate way of implementation and porting of the control application.

1 Mobile Interface To The Harness Metacomputing System

Harness is an experimental metacomputing framework based upon the principle of dynamically reconfigurable, networked virtual machines. Harness supports reconfiguration not only in terms of the computers and networks that comprise the virtual machine, but also in the capabilities of the virtual machine itself. These characteristics may be modified under user control via a *plugin* mechanism, that is the central feature of the system. The plugin model provides a virtual machine environment that can dynamically adapt to meet an applications needs, rather than forcing the application to conform to a fixed model [4].

The fundamental abstraction in the Harness metacomputing framework is the *Distributed Virtual Machine (DVM)*. Heterogeneous computational resources may enroll in a DVM at any time, however at this level the DVM is not yet ready to accept requests from users. To begin to interact with users and applications, the heterogeneous computational resources enrolled in a DVM need to be loaded as plugins. A plugin is a software component that implements a specified service. Users may reconfigure the DVM at any time both in terms of the computational resources enrolled, and in terms of services available by loading / unloading plugins [3].

The current prototype of Harness is a set of Java packages, which implement the functionality of the DVM, as well as some standard services (a farmer / worker model, the messaging subsystem and a PVM compatibility plugin) that allows a programmer to develop distributed applications with relatively little effort. The system is able to run in totally heterogeneous environment, as long as the nodes enrolled are connected to the network and can communicate with each other.

The recent advances in the field of hand-held computing are motivating designers and programmers to consider adding mobile interfaces to their systems. Hand-held devices are very specific, because of their limited resources and the way people use them

V.N. Alexandrov et al. (Eds.): ICCS 2001, LNCS 2073, pp. 367–374, 2001.
©Springer-Verlag Berlin Heidelberg 2001

(e.g. it can not be assumed that a device is permanently connected to the Internet). Therefore the mobile component of the system has to be designed with care so that the majority of processing is performed on the system side and not at the hand-held.

We postulate that there exists the following four main modes of operation involving mobile computers [6]:

- a single wearable computer
- a peer-to-peer wireless network of wearable computers
- a wearable computer and metacomputing system connected together
- a peer-to-peer network of metacomputing wearable computers

In terms of Harness, modes three and four are of most significance. In the third mode of operation, single users will connect to a back-end metacomputing system to conduct experiments, simulations or indeed any other activity that requires more processing power than is available at the hand-held [6]. Such powerful, heterogeneous computational back-ends can be provided by a Harness DVM. As it will be shown later in this document, the control of the DVM from a mobile device can be achieved with little additional implementation.

With the current capabilities of the hand-held devices and the development tools it's not possible to run a fully functional Harness kernel in such a limited environment. The first problem is the lack of a complete Java Virtual Machine for hand-held devices (there is a very limited virtual machine called KVM, but it's not powerful enough to run a Harness kernel). Further challenges are presented by the limited resources available. Therefore instead of extending Harness to hand-held devices, we propose a new design for a control application that is suited to mobile computers. Using such an application, a user of a hand-held device would be able to connect to the DVM, check it's state and modify some parameters. Due to technical issues (most notably object serialization and de-serialization used heavily by Harness), the application must have a 'proxy' at the Harness side. Thus, a specialized Harness plugin, which would translate requests from the application to the internal Harness protocol and vice versa is required. Since the application will be running on a limited device, the protocol between the mobile computer and the proxy plugin has to be as simple as possible. The remainder of this paper is structured as follows. Section 2 describes the application-plugin model more specifically, whilst in section 3 we present some conclusions about the possible functionality of this model.

2 Harness Control Application For Hand-Held Devices

As noted above, it is not possible with the current capabilities of hand-held devices to run a Harness kernel on them. Nevertheless, it is quite possible to implement a specialized application to control the DVMs state (i.e. to load / unload plugins, grab / release hosts, query the current state and intercept particular events). The model of such an application is a complementary pair of entities: a Harness plugin executing on a desktop computer and a small application running on the hand-held device. The plugin executes commands concerning the DVM on behalf of the user, who issues commands using a small application on the hand-held device. The main issue in this model is the communication between the application and the plugin. It is not profitable to implement specialized modules in the plugin, which would handle different types of communication (serial, modem, infrared, network), especially as each would require a significant

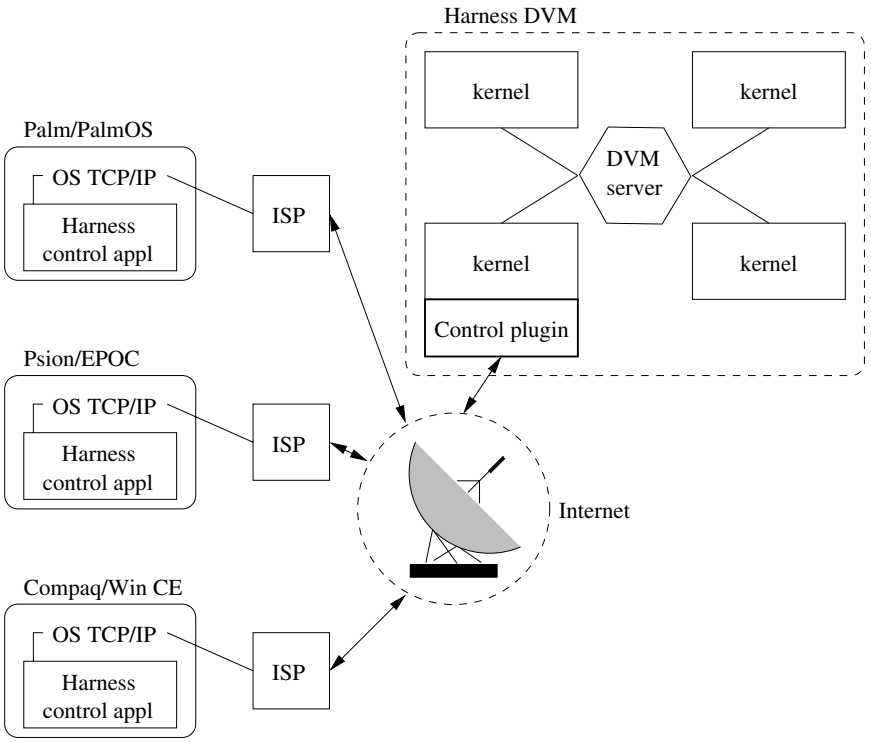


Fig. 1. The Harness Control Application Communication Model.

effort to implement. Instead, a more general solution is to implement the communication using the TCP/IP layer and the socket model. This strategy is utilized by most of the applications for hand-helds and offers the most general mechanism.

A TCP/IP connection is be provided by the operating system via an ISP (Internet Service Provider). This in turn assigns an IP address to the device. Thus, the control application itself does not depend on the type of the connection between the hand-held device and the ISP. Wired modem connections can be used as can GSM wireless connections or indeed any other bridging technology (e.g. infrared, LAN). Therefore, the principle of this model is that the type of Internet connection is completely transparent to the control application.

The protocol between the control application and the plugin should be extensible and heterogenous. In this instance, a simple text-based protocol is the most appropriate. A more sophisticated protocol can be introduced when experience with the system is gained. Thus, to avoid problems with differences in data representation, object serialization and several other technical issues, a text-based protocol will be adopted.

As an example exchange using the protocol, consider the following. The control application, after connecting to the plugin via a socket connection, sends the command LIST HOSTS and obtains a list of hosts as the answer. Other commands can be introduced (e.g. GRAB hostname, RELEASE hostname, LOAD plugin name hostname) to change the DVM status.

The implementation of the control application for hand-held devices is a pertinent issue. Since the devices, their operating systems and SDKs differ, there is no possibility to develop one application for all possible types of hand-held device. One can envisage such an application in Java, but Sun's Java 2 Micro Edition [2] (targeted specifically at hand-held devices) has very limited functionality and is still unavailable on some platforms. Therefore we propose to build native applications for different platforms exploiting software re-use to make migrating to a standardized approach trivial in the future. With the correct abstraction, it is possible to separate the system independent functionality of the application from the communication and visualization components, which are system specific.

Preliminary investigation explored the possibility of implementing a Harness extensions on the Palm Computing platform [5]. The prototype can be written in C using GNU-based PRC Tools (v. 2.0) for Palm and PalmOS SDK v. 3.5. The PalmOS platform contains a very convenient NetLib library that offers functionality similar to the Berkeley socket library. Although the data structures and function calls differ from the Berkeley version, it is possible to rapidly produce a working prototype using this library, for the purposes of testing and conducting further experiments. The NetLib library abstracts from the underlying transport layer rendering it suitable for use in this project. The application will use a system-level TCP/IP stack making the physical connection transparent. Thus, the Harness control application for PalmOS will connect to the Internet using any mechanism supported by the operating system.

It is impractical to create a fully functional Harness front-end for PalmOS without a significant amount of implementation effort. The difficulty stems from the nature of internal Harness communication. The Harness protocol relies on serialized Java objects being sent between the kernels and plugins. All Harness events are in fact Java objects and as such cannot be passed directly to a Palm (especially when they contain Java remote references, which are completely unmeaningful and useless on Palm). There are two types of events in Harness: system-generated and user-defined. The system-generated events form a relatively small set and can be translated by the proxy plug-in to a simpler form, understandable by the control application on the hand-held. The user-defined events can contain any serialized Java objects, thus there is no way to translate them into a form useful for the control application (assuming the application is not able to de-serialize and utilize Java objects).

Another problem with the Palm computing platform is that an application cannot be multithreaded. That implies a particular model of communication. An application can not be event-driven (in terms of Harness events) unlike the Harness Login GUI. Instead of waiting for an event to arrive from the proxy plug-in (which would block the whole application) the application has to actively query the plugin to obtain the recent system events. This can be achieved either by user request, or automatically, by downloading the list of events periodically. Both modes cause some unnecessary network traffic (the original Harness Login GUI receives the events only when they occur), but given the limitations of the system, this is unavoidable. The general model of a Palm application is shown in fig. 2 [1].

Figure 2 shows why there can not be a socket listening operation in the event loop. If a block is introduced in the 'event queue' step, then the whole application will block (as it will not receive any more system events). Although this is a technical limitation, it causes the application model to be constructed in a specific way i.e. the active querying described above must be adopted.

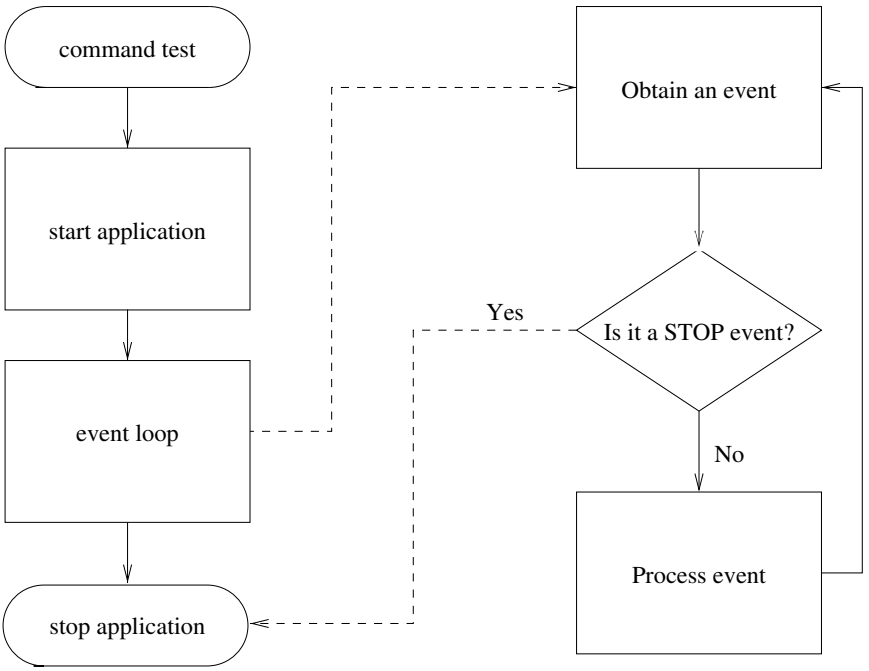


Fig. 2. General Model Of A Palm Application

3 Functionality Of The Harness Control Application

According to [6], the Harness control application represents an approach, in which a single mobile computer is connected to a heterogeneous computational back-end. Although fig. 1 shows more than one hand-held device connected to the Harness metacomputer, it is still a single mobile machine model because each of the devices shown is connected to the proxy plug-in independently. Thus, the users operating the hand-held devices may not have knowledge of each other. Therefore that is not simply a model with a network of wireless devices connected to a computational back-end.

M. Migliardi and V. Sunderam [6] proposed more a sophisticated model for connecting a wireless device to the Harness metacomputing system. This approach assumes a proxy plug-in is a fully functional Harness kernel, appearing to the rest of the Harness system as another node enrolled into a DVM. The hand-held device can communicate with the 'ghost' kernel and interact with the whole DVM using it as a proxy. Unfortunately the implementation of this model requires a significant investment of effort, because either the original Harness protocol must be used to provide a hand-held device with data, or all messages must be translated to a simpler form that is understandable by the hand-held device application. This approach can be easily implemented if a fully functional Java virtual machine existed for a particular mobile architecture, but as was noted above, the Micro Edition of Java 2 offers limited functionality and is still under development.

Thus, currently it is easier to add some extra functionality to the proxy plug-in and extend the hand-held-application-to-plugin protocol, than to implement a partially Harness compliant module for hand-held devices.

With a relatively small effort it is possible to implement a control application and a proxy plug-in. This will facilitate the following:

- query the status of the DVM (get a list of hosts, a list of plug-ins loaded at each host and a list of the hosts available to grab),
- grab and release hosts,
- load and unload plugins,
- intercept some of the events (those which can be translated to a non-object form and sent to the hand-held device),
- inject some user events (containing a string as payload).

This functionality offers a remote user good DVM control and limited control over simulations. Note, that there are only two ways a user can start / stop / modify a simulation. First, is by loading and unloading plug-ins. This method is not convenient as there is no way a user can provide or change the simulation parameters. The second method is by injecting some user events. This permits the inclusion of some parameters for the simulation and to control it more fluently, but the plugins, that do the simulation itself have to be written in a specific way to understand those events and act on them properly. The hand-held device can also receive some information from the plugins enrolled in the simulation. However, this information is usually simulation specific, and too verbose to be shown to the user in a readable form. Nevertheless, to control a very complicated simulation with large amounts of data, it is more appropriate to implement a specialized simulation front-end for hand-held devices, which would be able to represent the simulation status in a graphical form. Such an application would work in the same way as the DVM control application, but instead of connecting to a general DVM proxy and using a standard protocol, it would connect to a specific simulation control plugin and use the most convenient protocol.

4 Collaboration Using A Harness Proxy Plugin

Our investigation into the Palm computing platform shows, that other types of collaboration are possible, not only in terms of simultaneous simulation control, but also between the users themselves. The proxy plugin, described in previous sections, can be used also as a central point in the collaborative communication between the users of the hand-held devices. Moreover, this communication can also include users operating desktop computers, providing they are using appropriate front-ends.

The Harness event mechanism is a good tool to achieve this goal. Suppose the users connected to the proxy plugin want to talk to each other using their mobile computers. All the messages can be implemented as a special variety of user-generated Harness events. By extending the proxy-plugin is is possible for it to intercept and inject that type of events making them available for both users of the mobile computers and users of the desktop computers. Text messages are not the only type of data, which the application-proxy mechanism can manipulate. By adding new types of events users can send / receive bitmaps, sounds and other multi-media. The important issue is that the proxy plugin will keep the link with all of the connected mobile users, thus, it is not necessary to use a special server to cooperate with the other users. Moreover, as only people

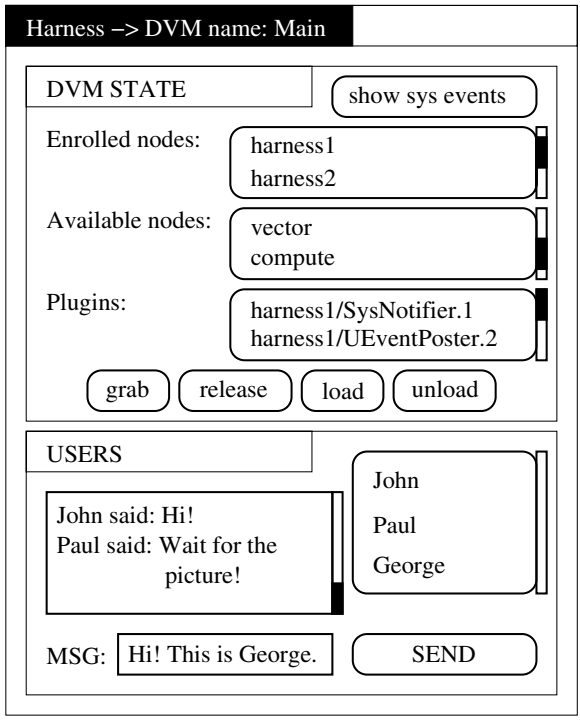


Fig. 3. The design of the control application GUI for Palms.

working on a common project will connect to a particular DVM they automatically will be able to communicate with their co-workers.

The proxy plugin can translate user-generated events of the specified type (user messages) into the format understandable by the control application. By extending the control application we can offer the users the ability to talk to each other. This brings the fourth mode of operation, described in section 1, into operation.

As an illustration, a possible GUI for the collaborative control application is shown in fig 3. The plugin itself, apart from the functionality described earlier, will perform the following activities:

- keep track of the users currently connected
- detect the additional types of events and take appropriate actions (translate the event's payload and send it to the message recipient or to all users if it is a broadcast message)
- queue the user messages for each connected user separately (since the messages will be sent to the user only when the control application asks for them, thus it is possible that there are more than one message present between such requests).

5 Conclusions

The Harness control application for mobile devices is a powerful extension of the Harness metacomputing system. The ability to check and change the DVM state and steer

the simulation from different locations is a desired feature. Unfortunately the Java 2 Micro Edition architecture in it's current development and hardware support stage is not powerful enough to implement all of the features mentioned in [6]. Although facilities exist in the Java 2 Micro Edition distribution to handle basic network connections, it is not possible to serialize / de-serialize objects, which is a required feature to implement a fully functional Harness extension. Nevertheless, the implementation of a simpler control application for mobile computers (the one described in the previous section) would be particularly useful in terms of collaborative computing, and this can be implemented on most hand-held computers using their native SDKs. Providing that the application is designed to exploit the benefits of software re-use, it can be easily ported to the different platforms by re-writing the communication and visualization modules. The use of a simple text-based protocol guarantees that all platforms, which can handle network connections, will also be able to communicate using this protocol.

In the near future the control application for the PalmOS platform will be implemented. It can then be ported to the Psion/EPOC and Windows CE platforms. This would comprehensively cover the majority of the hand-held computer market. Simultaneously, an alternative version in Java can be produced, since it is likely that Java 2 Micro Edition will also develop in parallel.

The prototype at first will handle only the DVM state tracking / control. At later stages, the extended user collaboration functionality can be added to both the application and the plugin, as described in section 4. The complete collaborative control application can then be ported to different platforms and the Java front-end for desktops workstations can be implemented.

References

1. A. Howlett and T. Tso and R. Critchlow and B. Winton. *GNU Pilot SDK Tutorial*. Palm Computing Inc.
2. Curtis Sasaki. *Java Technology And The New World Of Wireless Portals And M-commerce*. Sun's Consumer Technologies, Sun Microsystems. available from: `http://java.sun.com/j2me/docs/html/mcommerce.html`.
3. M. Migliardi and V. Sunderam. Heterogenous Distributed Virtual Machines In The Harness Metacomputing Framework. In *Proc. of the Eigth Heterogeneous Computing Workshop*, April 1999.
4. M. Migliardi and V. Sunderam. The Harness Metacomputing Framework. In *Proc. of the Ninth SIAM Conference on Parallel Processing for Scientific Computing*, March 1999.
5. Palm Computing Incorporated. *PalmOS Software Documentation*. available from: `http://www.palmos.com/dev/tech/docs/`.
6. V. Sunderam and M. Migliardi. Mobile Interfaces To Metacomputing And Collaboration Systems. Technical report, Emory University, 2000. A Proposal To The NSF.

Flexible Class Loader Framework: Sharing Java Resources in Harness System

Dawid Kurzyniec and Vaidy Sunderam

Emory University, Dept. of Math and Computer Science
1784 N. Decatur Rd, Atlanta, GA, 30322, USA
{dawidk, vss}@mathcs.emory.edu

Abstract. Harness is a Java-centric, experimental metacomputing system based on the principle of dynamic reconfigurability, not only in terms of the computers and networks that comprise the virtual machine, but also in the capabilities of the VM itself. These characteristics may be modified under user control via a "plug-in" mechanism that is the central feature of the system. This "plug-in" mechanism relies on the underlying class loader services which enable loading and accessing Java classes and resources from a set of software repositories. However, the specific class loading requirements imposed by Harness cannot be satisfied by any of the class loaders provided as a part of Java platform. In this paper we describe new, flexible, class loading techniques which are primarily intended to improve plug-in management and security aspects in the Harness system but can also benefit other applications, especially those collaboratively accessing shared Java resources.

1 Introduction

Harness [5,11] is a metacomputing framework based upon such experimental concepts as dynamic reconfigurability and extensible virtual machines. The underlying motivation behind Harness is to develop a metacomputing platform for the next generation, incorporating the inherent capability to integrate new technologies as they evolve. The first motivation is an outcome of the perceived need in metacomputing systems to provide more functionality, flexibility, and performance, while the second is based upon a desire to allow the framework to respond rapidly to advances in hardware, networks, system software, and applications.

Harness attempts to overcome the limited flexibility of traditional software systems by defining a simple but powerful architectural model based on the concept of a software backplane. The Harness model is one that consists primarily of a kernel that is configured by attaching "plug-in" modules that provide various services.

As is natural for a Java-centric system, "plug-in" modules are basically sets of related Java classes and possibly other resources cooperating to provide the desired service. Thus, the ability of Harness to attach "plug-in" modules is based upon the underlying class loader's capabilities to locate and load classes from a dynamically changing set of software repositories. The current implementation

V.N. Alexandrov et al. (Eds.): ICCS 2001, LNCS 2073, pp. 375–384, 2001.
© Springer-Verlag Berlin Heidelberg 2001

of the Harness class loader fails to satisfy two desired features: supporting loading classes from Java archive (JAR) files, and supplying a uniform and reliable caching mechanism to reduce potentially expensive network access.

This paper describes the design and implementation of a flexible class loader framework and its application in Harness system. Although initially designed to support Harness, it is generic and powerful enough to be successfully used in other applications, which include, but are not limited to, the systems collaboratively accessing shared and dynamically changing Java resources. The remainder of this paper is organized as follows. Section 2 provides a general discussion of the role of class loaders in the Java language. Section 3 presents related work. In Section 4 we describe the architecture of the flexible class loader framework; its intended application to the Harness system is presented in Section 5. The paper concludes (Section 6), with an outline of future work plans.

2 Class Loaders in Java

The concept of a class loader evolved together with the Java technology. Primarily it was designed to allow users to load Java VM classes from various non-standard sources such as those downloaded from the network, fetched from a database, or even generated on-the-fly. In JDK 1.0.x, with the API[1] consisting of only four methods loading classes was its only purpose. JDK 1.1 incorporated extensions to the class loader to find and load additional resources such as images, icons, sounds, and property files, which could accompany classes used in an application. Other improvements included an an attempt to extend the Java security model by introducing class signers.

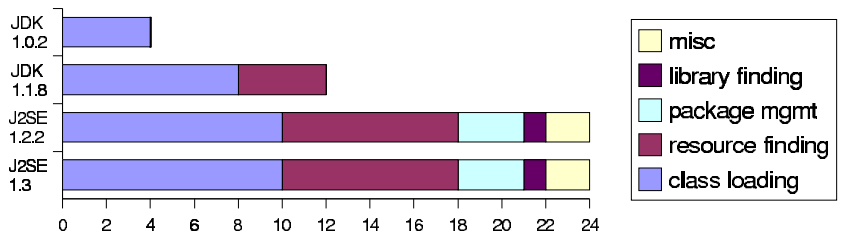


Fig. 1. Class Loader API evolution

The Java 2 platform brought substantial changes to class loader capabilities. The semantics were refined [10] and the design has been tightly integrated with the newly established Java Security API [12]. Two new

[1] Thorough this paper, we use term "class loader API" to denote `public` and `protected` methods of the class `java.lang.ClassLoader`.

classes were added: `java.security.SecureClassLoader`, intended as a base for user-defined class loaders taking advantage of Java security model, and `java.net.URLClassLoader`, which enables loading data from Java archive (JAR) files [7] and includes support for downloadable extensions [2]. Moreover, in addition to finding and loading classes and resources, the class loader became responsible for defining and keeping track of packages and for controlling native library linking. Additionally, the notion of parenthood and delegation was introduced for class loader objects: every class loader is encouraged to delegate class loading requests to its parent before it tries to load classes by itself. The class loader API in Java 2 consists of 24 methods as compared to 4 methods in the original version, highlighting the scale of changes that class loaders have undergone. The evolution of the class loader API is illustrated in Fig.1.

Prior to Java 2, native libraries were almost completely handled in native VM code. This was restrictive to library management: to be loaded by the VM, libraries had to be present in some VM-specific, designated place, typically somewhere in the local file system (e.g. within LD_LIBRARY_PATH on UNIX platforms). Applications which required more complex library management such as dynamic download were forced to adapt sophisticated approaches like preliminary static analysis of the bytecode of a class in order to scan for native library dependencies [4], or use security managers merely for code downloading purposes.[2] In Java 2, native libraries are loaded by class loaders rather than by the Java VM itself. Much of the native library handling code was rewritten in Java giving developers an opportunity to handle library lookup requests in an application-specific way, which provides more flexibility in native library management.

3 Related Work

The `java.lang.URLClassLoader` is a class loader introduced in Java 2 to address loading classes and resources not only from ordinary directory structures but also archived in JAR files. It maintains a search path consisting of a set of base URLs pointing either to directories or JAR files and it searches for classes and resources using this path. It also understands downloadable extensions, so it allows JAR files to add URLs of other JAR files that they depend on to its search path. Such a scheme fits well the industrial software distribution model where code is bound tightly to an organization providing the software, but does not necessarily fit other models, e.g. when the software is stored in large repositories and accessed dynamically as in case of Harness system. The `URLClassLoader`'s class and resource searching algorithms are fixed, making it inappropriate as a base class for implementing such distribution models. The Java platform thus lacks a class loader which would be both flexible and JAR-aware. Also, no class

[2] Harness did this using `SecurityManager.checkLink(String)` method; see `http://java.sun.com/j2se/1.3/docs/api/java/lang/Runtime.html#load(java.lang.String)`.

loader-related proposal can be found in the Java Specification Requests area [9] in Sun's Java website.

Although there are many systems providing customized class loaders to satisfy application-specific requirements [3,6], we do not know about any efforts to develop an architecture that would be generic and customizable enough to be reusable across different systems.

4 Flexible Class Loader Framework

Class loaders are used when there is a need for some non-standard, application-specific way to find classes, resources, or libraries. Nevertheless, such specialized class loaders often have much in common. For example, common tasks might include reading data from JAR files or downloading classes from the network; similarly, many class loaders do not have separate routines for dealing with classes and resources but access them in a uniform way. These similarities, however, do not comprise a common denominator. Therefore, full advantage can be taken of a generic class loader only when its structure is *modular*, so that users can choose and configure the modules they require in their applications.

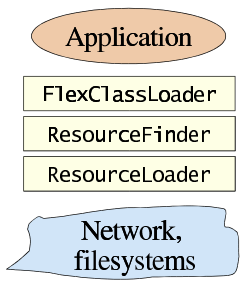


Fig. 2. Structure of Flexible Class Loader Framework

The flexible class loader framework described in this Section presents an example of such a modular structure, as illustrated in Fig.2. It is based on the observation that the typical class loader, apart from carrying out its low level function, must perform independent activities like searching for requested data, downloading data and classes from the network and managing JAR files. Taking this into account, it is reasonable to split the functionality into three layers:

- the class loader itself, serving as an interface to an application,
- the resource finder, encapsulating the searching algorithm but unaware of differences between classes, libraries, and other resources,
- the resource loader, responsible for managing JAR files and downloading data from the network.

The most specific part of each class loader is its resource searching algorithm. For that reason, many sophisticated class loaders can be derived from this scheme just by customization of the resource finder layer. In the following subsections, we describe all three layers in more detail.

4.1 `FlexClassLoader`

The `FlexClassLoader` class is the class inherited from `SecureClassLoader`. Its functionality its intentionally similar to that of `java.net.URLClassLoader` – the code for defining classes and packages is virtually the same – though it is intended to be generic and customizable. It does not perform data searching operations, but relies instead on the capabilities of its underlying resource finder (provided as a constructor argument).

The flexible class loader framework introduces the notion of *resource handles*.[3] A resource handle is a set of meta-information related to some resource, regardless of whether it is a Java class, native library, image, text file or any other entity. The resource handle provides such information as the resource name, its URL (and origin URL in cases when resource was cached), the URL of its code source (and its origin version as appropriate), and finally the certificates and attributes of the resource as well as the JAR manifest in case the resource is from a JAR file. Furthermore, the resource handle allows loading of the resource, either through the provided input stream or directly into the `byte[]` array. Note, however, that the handle does *not* include the resource; rather, it represents a connection to the resource – and there is no guarantee that resource loading will succeed after the handle is obtained. The `FlexClassLoader` has four designated methods enabling its subclasses to take advantage of resource handles:

```
protected ResourceHandle findClassHandle(final String name);
protected ResourceHandle findResourceHandle(final String name);
protected Enumeration findResourceHandles(final String name);
protected ResourceHandle findLibraryHandle(final String name);
```

The `FlexClassLoader` assumes that classes, resources, and libraries should be searched for in the same way. Its methods related to class and library loading simply transform the class or library names to resource names and perform generic resource lookup using single resource finder, as described in the next Section. If this is not appropriate for some particular implementation that wishes to handle classes, resources, and libraries in a specific way, these methods may be overridden to provide the desired functionality, perhaps using designated resource finders.

4.2 `ResourceFinder`

The resource finder is the entity related to the most specific feature of each class loader – the resource lookup algorithm. In the flexible class loader framework, the entity is represented by the `ResourceFinder` interface, which should be implemented by users to provide a specific searching approach:

[3] The concept is based on the class `sun.misc.Resource`.

```
public interface ResourceFinder {
    /** Returns ResourceHandle of resource with specified name
     *  or null if the resource cannot be found. */
    ResourceHandle findResource(String name);
    /** Returns enumeration of ResourceHandle objects representing
     *  all resources with specified name. */
    Enumeration findResources(String name);
}
```

An important feature of this layer of the framework is that it does not distinguish between types of resources it is asked to search for – resource finders handle only simple, unified queries. Therefore, this two-method interface is sufficient to handle all requests of the `FlexClassLoader`.

A number of customized class loaders are being designed to support loading data from the network. To limit the overhead of accessing data through a potentially slow network connection, it is often desirable to provide a caching mechanism which would allow the storage of downloaded data on the local file system. To address this issue, the flexible class loader framework provides the abstract class `CachingResourceFinder`. To take advantage of this class, users must implement its two abstract methods:

```
public abstract class CachingResourceFinder
    implements ResourceFinder
{
    ...
    protected abstract ResourceLoader getResourceLoader();
    protected abstract Iterator getSearchEntries(String name);
    ...
}
```

The first method should return an instance of the `ResourceLoader` object (as described in next Section) intended to manage resource loading. By implementing the second method, users specify an actual resource lookup algorithm, providing the iteration over consecutive URLs at which the given resource should be subsequently searched for. Both the `findResource()` and `findResources()` methods needed to implement the `ResourceFinder` interface are implemented in `CachingResourceFinder` and they assume that the resource lookup algorithm is simple enough to fit in the scheme of a series of URLs to search for. For more sophisticated lookup algorithms users can override those methods and still take advantage of low-level caching functionality provided by this class.

The caching policy implemented in the `CachingResourceFinder` uses the designated cache directory where it replicates the remote directory structures as resources are loaded from the network. When the specific resource is searched for, the search iterator returned by the `getSearchEntries()` method is iterated twice: in the first pass, the URLs are converted to their cache equivalents so the resource is searched for in the cache. The second pass goes through original URLs; if the resource is found, it is downloaded and stored in the cache for future use.

4.3 `ResourceLoader`

The bottom layer of the flexible class loader framework is represented by the `ResourceLoader` class:

```
public class ResourceLoader {
    ...
    public ResourceHandle findResource(URL base, String name) { ... }
    ...
}
```

This class is a uniform and convenient way to load resources from the file systems and the network, and additionally provides support for JAR files; this support includes attaching appropriate meta-information to resource handles as the meta-information is fetched from the JAR file manifest.

Handling JAR files requires special care. On one hand, the process of decompressing a JAR file, which often involve verification of digital signatures, can be time consuming, so it is reasonable to keep JAR files opened for some period of time in order to speed-up subsequent resource fetching. On other hand, however, opened JAR files can consume a lot of system memory, so they should be closed as soon as possible. The `ResourceLoader` tries to balance these contradictory goals as follows. It keeps the specific JAR file opened as long as there are any live resource handles representing resources loaded from it. When there are no more such handles, instead of being immediately closed the JAR file is moved to a special, limited-size cache pool. The maximal size of the cache pool may be specified by the user and it can be as low as 0 (so that JAR files are closed immediately when they are no longer referenced by resource handles) or may be set to infinity (what means that JAR files, once opened, are never closed).

4.4 Cache Coherency Issues

Perhaps the most important challenge for any caching mechanism is to retain cache coherency. This issue may be addressed in numerous ways, depending on the level of system integration and how strict the coherency requirements are. In the flexible class loader framework we adapted the simple expiration-time oriented algorithm, similar to the one used by HTTP proxies. The `CachingResourceFinder` object has an associated validity-duration value, which denotes how long a given cache entry is considered valid after it has been created. In addition, if the cached entry is a JAR file, it can explicitly override this value providing its own Expiration-Time attribute in a JAR Manifest.

Another issue is related to the JAR files kept open by the `ResourceLoader`. As long as the JAR file remains open, it ignores modifications and even deletion of the actual file from which it was constructed, so the application is not notified that the file changed. To address this issue, the `ResourceLoader` class provides the `invalidate()` method used to notify it that the given JAR file is no longer valid and should be closed. Depending on an additional argument, this either closes the file immediately and invalidates all resource handles derived

from it, or it marks the file to be closed as soon as it is no longer referenced by any handles, thus providing an opportunity to be smoothly released. The `CachingResourceFinder` class provides three cache coherency levels based on that feature. At the level HIGH, a change in the cached JAR file causes notification of the loader and immediate invalidation of all derived resource handles. At the level MEDIUM, the JAR file is marked to be closed after the handles have finished. At the level LOW, the loader is never notified about the change so the JAR file can remain open for unspecified periods of time, dependent on loader caching policy.

5 `FlexClassLoader` in Harness Metacomputing Framework

5.1 Hierarchical Resource Lookup

Harness uses a specific class loading scheme: classes can be stored in the set of software repositories, which can be maintained by independent organizations. Each time the class is to be loaded, it is subsequently searched for in the repositories. As long as all classes are stored as an ordinary files and there is a mapping between file paths and package names, they can be easily found just by constructing appropriate lookup URLs (this is the manner in which the current Harness class loader is organized). The only activity required to make a given class visible to Harness is to put it in appropriate place in the repository.

This model gets complicated if we wish to incorporate JAR files as an option to store classes and resources. It is not obvious how JAR files should be stored in repositories to retain simplicity of making the classes and resources available to the Harness system. Most Internet protocols (including HTTP as an important special case) do not provide reliable mechanisms to query for directory contents, so the JAR files must be stored in places where the class loader expects them to be found.

As a solution, we provide the notion of hierarchical resource lookup. It is based on the package-oriented organization of the Java language and the resulting fact that JAR files are encouraged to store single Java packages, which are sets of highly related classes cooperating to provide some service. Making such an assumption a requirement, it becomes possible to establish a contract for the class loader: a JAR file should have the same name as the last part of the package name and it should be stored in the repository in a place where that package is expected. For example, the JAR file containing the package `edu.emory.mathcs.harness` should be named `harness.jar` and be placed on the `edu/emory/mathcs` path in repository.

Fig.3 shows two examples of hierarchical resource lookup, based on the naming contract described above. The algorithm goes through the directory structure, trying to find the class or resource in the JAR file related to the outermost package possible. Finally, it looks for the JAR file containing the class or resource only, and if this fails it tries to load the class or resource in the traditional way. This algorithm is implemented in the `RepositoryResourceFinder` class, which

```
looking for harness.plugins.pkg1.Cls1:

harness.jar
harness/plugins.jar
harness/plugins/pkg1.jar

looking for
harness.plugins.pkg2.subpkg2.Cls1:

harness.jar
harness/plugins.jar
harness/plugins/pkg2.jar
harness/plugins/pkg2/subpkg2.jar
harness/plugins/pkg2/subpkg2/Cls1.class.jar
harness/plugins/pkg2/subpkg2/Cls1.class
```

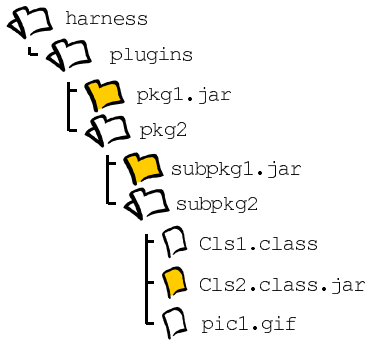


Fig. 3. Examples of Hierarchical Resource Lookup

extends `CachingResourceFinder` and inherits its caching capabilities. As a result, the exact structure of remote repositories is replicated in the local cache as data is loaded.

5.2 Benefits for Harness

In the Harness system, the class loader plays an important role as it is responsible for providing the background layer for linking "plug-in" services. The new class loading approach will be beneficial for Harness as it takes advantage of JAR files and provides a generic caching mechanisms:

- JAR files can be digitally signed and can contain certified code. This will allow security to be improved in Harness, as administrators will be able to specify security policies based on code certificates.
- JAR files are compressed, so data will be downloaded faster from the network.
- JAR files may contain meta-information related to the JAR file as a whole or to its separate entries. Using this meta-information, Harness will be able to avoid performing complex byte-code analysis of the classes being downloaded (the system performs such analysis in order to decide if it can run the "plug-in" in its own Java VM or if it should spawn another one).
- Complex plug-ins, which contain a number of classes and resources, will be maintained more easily by storing them in a single JAR file. Also, package consistency and versioning will be improved by taking advantage of JAR package attributing and sealing features.

6 Conclusion and Future Work

This paper describes the flexible class loader framework, which simplifies development of application-specific class loaders. The framework consists of a set

of customizable classes providing commonly needed functionality. Its intended application in the Harness system is also presented, including the hierarchical resource lookup algorithm supporting loading resources stored in software repositories.

Future work plans include enabling Harness to take advantage of "plug-in" modules containing native code [1,8]. To retain high portability, these modules are intended to contain native source files rather than precompiled libraries; the compilation process would occur "just in time" on the target platform. This technique would benefit from the described class loader framework in several ways: first, the Java classes and native source files comprising the "plug-in" could be bundled together in the JAR file. Next, the code certification and digital signatures could be used to introduce trusted code, which is essential when native code is considered. And finally, the general-purpose caching services would support management and caching of native libraries once they were compiled on a target platform.

References

1. M. Bubak, D. Kurzyniec, and P. Łuszczek. Creating Java to native code interfaces with Janet extension. In M. Bubak, J. Mościnski, and M. Noga, editors, *Proceedings of the First Worldwide SGI Users' Conference*, pages 283–294, Cracow, Poland, October 11-14 2000. ACC-CYFRONET.
2. The Java extension mechanism. `http://java.sun.com/j2se/1.3/docs/guide/extensions/`.
3. P. Gray and V. Sunderam. IceT: Distributed computing and Java. *Concurrency, Practice and Experience*, 9(11), Nov. 1997. Available at `http://www.mathcs.emory.edu/icet/IceT5.ps`.
4. P. Gray, V. Sunderam, and V. Getov. Aspects of portability and distributed execution of JNI-wrapped code. Available at `http://www.mathcs.emory.edu/icet/sp.ps`.
5. Harness project home page. `http://www.mathcs.emory.edu/harness`.
6. M. Izatt and P. Chan. Ajents: Towards an environment for parallel, distributed and mobile Java applications. In *ACM 1999 Java Grande Conference*, San Francisco, California, June 12-14 1999. Available at `http://www.cs.ucsb.edu/conferences/java99/papers/13-izatt.ps`.
7. Java archive (JAR) features. `http://java.sun.com/j2se/1.3/docs/guide/jar/`.
8. Java Native Interface. `http://java.sun.com/j2se/1.3/docs/guide/jni/`.
9. Java Community Process Java Specification Requests. `http://www.java.sun.com/aboutJava/communityprocess/`.
10. S. Liang and G. Bracha. Dymanic class loading in the Java Virtual Machine. Available at `http://www.java.sun.com/people/gbracha/classloaders.ps`.
11. M. Migliardi and V. Sunderam. The Harness metacomputing framework. In *Proceedings of the Ninth SIAM Conference on Parallel Processing for Scientific Computing*, San Antonio, Texas, USA, March 22-24 1999. Available at `http://www.mathcs.emory.edu/harness/PAPERS/pp99.ps.gz`.
12. Java security home page. `http://java.sun.com/j2se/1.3/docs/guide/security/`.

Mobile Wide Area Wireless Fault-Tolerance

J. S. Pascoe†, G. Sibley‡, V. S. Sunderam‡ and R. J. Loader†

†Department of Computer Science
The University of Reading
United Kingdom
RG6 6AY
J.S.Pascoe@reading.ac.uk
Roger.Loader@reading.ac.uk

‡Math & Computer Science
Emory University
Atlanta, Georgia
30302
gsibley@emory.edu
vss@mathcs.emory.edu

Abstract. This paper presents work-in-progress that is developing a novel fault-tolerant mechanism for use in mobile wide area wireless networks. As a developmental platform, we are using the Ricochet service which offers ubiquitous metropolitan scale wireless network coverage in several major US cities. We postulate that the majority of network failures in infrastructures such as Ricochet are caused by environmental factors. From this, we propose a GPS based mechanism that intelligently gathers semantic data pertaining to specific geographic areas (or *trouble spots*) which cause communication problems. To facilitate the categorisation of trouble spots, we propose a list of suitable metrics to analyse the status of a wireless connection. Finally, we experimentally evaluate their effectiveness.

1 Introduction

Recent work in the field of mobile wireless communication protocols has centered around *ad hoc* networking, in particular, disciplines such as routing have become very mature. Protocols such as ODMRP (On Demand Multicast Routing Protocol) [12, 11, 13, 6], AMRoute [1], CAMP [4], RBM [2] and LAM [10] all support this observation. Until recently, metropolitan scale wide area wireless research was not feasible as there were no suitable infrastructures to support such work. With the advent of developments such as Ricochet [9] and Bluetooth [7], this view is changing.

Recent work by the authors [19, 16, 15, 18, 17] has addressed the issue of fault-tolerance in multicast collaborative computing environments, in particular, the *Collaborative Computing Frameworks* [21] which has been developed at Emory University. Following the successful completion of this work, it was decided to extend the fault-tolerance protocols into the domain of wide area wireless communication. The exemplar for this work is the Ricochet wireless network which is rapidly becoming a popular city scale system for this type of research.

Ricochet is a wireless metropolitan area network that has been installed in several major US cities. It is a subscriber based system that offers a fixed bandwidth limit of 128Kbps through small portable radio modems. Note that unlike most other wireless networks, Ricochet is not ad hoc, that is, although nodes may move arbitrarily throughout the network, the communication infrastructure and thus the networks topology does not change. This requires that the importance of issues such as routing and intermittent communication be readjusted. Thus, in order to provide fault-tolerance within such a network, a new series of challenges must be addressed. The most notable of these, is

V.N. Alexandrov et al. (Eds.): ICCS 2001, LNCS 2073, pp. 385–394, 2001.
©Springer-Verlag Berlin Heidelberg 2001

the networks resilience in relation to environmental factors (e.g. a user may experience a temporary drop in communication whilst driving through a tunnel).

In this paper, we present a GPS based scheme that addresses these issues and also proposes a set of metrics to evaluate the state of a wireless connection. This in turn allows us to identify and categorise problem areas (or *trouble spots*), the underlying premise being to pre-emptively eliminate problems when other hosts subsequently visit the area.

This paper is structured as follows: in sections 2 and 3 we respectively describe our recent work in the Collaborative Computing Frameworks and outline the structure of the Ricochet network. Section 4 then presents our novel approach to fault-tolerance that extends the work discussed in section 2. To facilitate data collection within trouble spots, section 5 proposes a series of link-state evaluation metrics. In order to prove their effectiveness, we experimentally evaluate these metrics before finally giving our conclusions.

2 The Collaborative Computing Frameworks

The *Collaborative Computing Frameworks (or CCF)* is an environment that provides an Internet scale *collaboratory*. Highlights of the CCF include multiple levels of Quality of Service semantics, a novel variant on the virtual synchrony [21] algorithm, IP based WAN tunneling, an X multiplexor and a suite of purpose designed tools.

The underlying abstraction of the CCF is split into two distinct components; *sessions* are heavyweight groups, that is, each user wishing to participate in the collaboratory must register with a session. Once part of a session[1], each user is free to join an arbitrary number of *channels* or lightweight groups. Channels provide a mechanism for logically separating messages at different levels of Quality of Service. As an example, consider a collaborative audio tool that requires two channels to operate (the first for control messages, and the second as a data stream). The first channel may operate using a reliable mechanism whereas the second may be configured for unreliable data transfer (thus maximising data throughput). This abstraction allows an application to simultaneously operate across multiple levels of Quality of Service and in doing so, provides a more flexible transport mechanism than other collaborative environments.

Our recent work [19, 16, 15, 18, 17] addressed the issue of adding fault-tolerance into the CCF. Our approach differed from other datagram oriented schemes in that it is based on a reliable multicast primitive. This provides a significant enhancement over other work, since the number of message exchanges required to form consensus is drastically reduced.

The approach is based on two distributed elections. The first ratifies membership issues (such as failed hosts) and the second deals with the more subtle channel oriented faults. On detection of a failure, an election is called by the *error master* (i.e. the session owner). Each session member then probes the network to obtain an up-to-date snapshot of the sessions state. These results are then sent to the error master which collates, counts and multicasts the outcome of the election to the remaining session members[2].

The implementation of the mechanism is multi-threaded with one thread acting as an *error monitor* and the other as an *error handler*. The error monitor is responsible

[1] Note that the first user to join a session is called the *session owner*.

[2] For brevity, we do not discuss other scenarios such as the failure of the error master here. Instead, the interested reader is referred to [19, 16, 15].

for logging failures and mediating the invocation of the error handler which then coordinates the protocol's election phase. By basing the implementation on two complementing entities, the mechanism as a whole becomes more extensible, efficient and effective.

3 The Ricochet Network

The Ricochet network is present in 8 major US cities with further expansion into another 13 being planned in the near future. The network offers a fixed bandwidth of 128Kbps and is accessible to the general public through small portable wireless modems. The network itself consists of 7 architectural components which are grouped hierarchically to provide the service.

- *Wireless Modems* – Ricochet modems are small devices that can be connected to any portable workstation.
- *Microcell Radios* – Microcell radios communicate with user workstations through wireless modems. In addition, they perform error checking and are responsible for sending acknowledgements.
- *Wired Access Points* – Wired Access Points or WAPs are installed within a 10-20 square mile area. Each WAP collects and converts wireless data into a form suitable for transmission on a wired IP network.
- *Network Interface Facility* – A Network Interface Facility connects a group of WAPs to the Network Operations Centers.
- *Ricochet Gateway* – The Ricochet gateway is part of each Network Operations Center and connects the Ricochet network to other systems.
- *Network Operations Centers* – The Network Operations Centers are the hub of the entire system. They provide a means for monitoring and control within the network.
- *Name Server* – The name server maintains access control and routing information for every radio and service within the wireless network.

The overall topological structure of the networks wireless component is a mesh, since microcell radios are placed at regular intervals in a 'checkerboard' fashion. In [12] the authors argue that the additional redundancy in a mesh provides a more robust wireless network topology than that of a tree or a star.

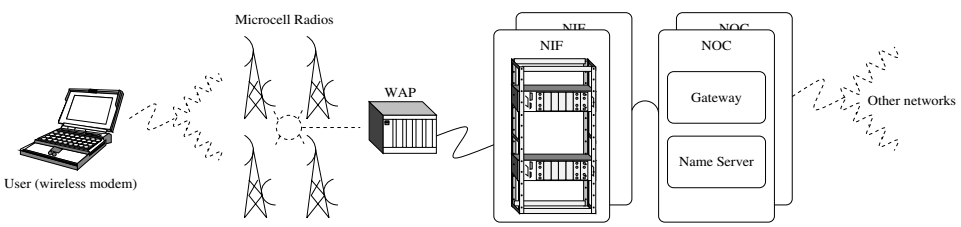


Fig. 1. The Ricochet Communication Architecture

4 Wireless Wide Area Fault-Tolerance

The migration to a wireless transmission medium in the CCF requires the consideration of several new fundamental issues:

- *Message Garbling* – Messages are more likely to be garbled, particularly in noisy environments (e.g. the central area of a city). In terms of the CCF, this has implications in guaranteeing reliable Quality of Service semantics.
- *Intermittent Connectivity* – Hosts are likely to experience unpredictable intermittent connectivity problems that can last for arbitrary amounts of time. In this situation, a loss of contact should not result in the host being removed from the session (if at all possible).

As noted above, we consider the majority of problems in a Ricochet network to be caused by environmental factors. Preliminary experimentation with the system has revealed that specific types of problem occur in relation to geographic locations (trouble spots). From this, we propose the following classification of trouble spots in a metropolitan area wireless network (see table 1). For brevity, a comprehensive classification is not given here. A discussion on additional metrics such as signal strength is given in [22].

Note that this classification also extends to failures at a network level. For example, the failure of a microcell radio would either be classified as a fatal disconnection or a long disconnection (depending on whether the user was stationary or moving).

Trouble spots can be of an arbitrary size, but often occur in predictable patterns. Trouble spots of the first classification would include areas in which there are many high buildings present (e.g. the center of a city). Short disconnection trouble spots are typically small and span an area of less than 10 square meters. Long disconnection trouble spots are inherently larger and are exemplified by road bridges and tunnels. Fatal disconnection areas are small, specific and occur in regular formations (e.g. lines). This type of trouble spot is usually encountered when a wireless modem moves beyond the networks range, for example, as the user travels out of the city. Using Ricochet, the user experiences a sharp drop in communication i.e. fading does not occur.

In order to facilitate mobility, the error monitor / error handler protocols require extension to incorporate an intelligent geographic view of the network. Composition of this view can be achieved by collecting data about trouble spots as they occur and then using this information to prevent (or intelligently deal with) other hosts encountering the same problem. The worst case scenario is when the trouble spot is not known to the system. In this instance, the following chain of events is observed:

No.	Description	Characteristics
1	Message garbling	messages are garbled or echoed
2	Short disconnection	suspended connection (short period)
3	Long disconnection	suspended connection (longer period)
4	Fatal disconnection	connection dropped

Table 1. Classification of Trouble Spots in Metropolitan Area Networks

1. A user encounters an arbitrary communication problem as they are moving across a city. This problem can be of any nature (e.g. a link failure or transmission errors). The nature of the trouble spot is identified and the geographic position is noted.
2. A server (running as part of the CCF) is contacted and informed of the new trouble spot. The trouble spot is categorised and some appropriate resolution strategy is adopted.
3. The host performs any action necessary to rectify the situation.

In the second scenario, the user encounters a known trouble spot and so the situation is dealt with differently. The server realises that a host is about to encounter a network problem, and so depending on the category and the semantics of the trouble spot, it can respond in a number of ways:

- *change the channel QoS dynamically* – the host maybe traveling into an area that is known to garble messages. In this case, the server may request that the QoS for any unreliable channels be upgraded to a reliable mechanism until the host moves out of the affected area.
- *do nothing* – the trouble spot may cause a small problem (e.g. a short disconnection) that can easily be accommodated by the buffering in the existing protocols.
- *begin logging messages* – a longer drop in communication is too substantial for the existing protocols to deal with. However, it is not usually necessary to forcefully remove the session member and so the server may begin to record messages to later facilitate an off-line replay.
- *warn the user* – the host maybe traveling beyond the range of the network and so a fatal disconnection may be about to occur. In this case the server may warn the user of the impending problem and ask for an indication of intent. If the user opts to leave the network, then the error master will forcefully remove the participant to preserve the sessions health.

Regardless of whether the host has prior information about a trouble spot, it must re-examine the area to ascertain whether its semantics have changed. If this is the case, the server is updated with the new information. Note that an individual server will store trouble spot data for a particular area. Mechanisms for distributing such servers are being investigated and will be reported on in a future presentation.

The scheme outlined above requires that a node periodically sends its location to the server. For this, we use the Global Positioning System (GPS). The idea of using GPS as a protocol enhancement is not new; ODMRP [11] makes extensive use of GPS in its 'mobility prediction' system. Also, the use of GPS for gathering positional data in wireless networks is mentioned in [12]. However, as we intend to use GPS in a fault-tolerant context, there is (at least to the authors knowledge) no conflict in novelty.

GPS offers a significant number of advantages over a network dependent (triangulated) mechanism. Most notably, GPS offers complete worldwide compatibility and will deliver positional information to an accuracy of within 10 meters. In terms of a metropolitan area network, this granularity is considered to be sufficient for most applications. The fault-tolerance mechanisms presented here will operate in any wireless environment and exhibit no dependencies on both the network and the CCF.

5 Wireless Link-State Evaluation Metrics

One outstanding issue is the pertinent question of how to ascertain the semantics relating to a specific trouble spot. When encountered, a trouble spot is initially detected

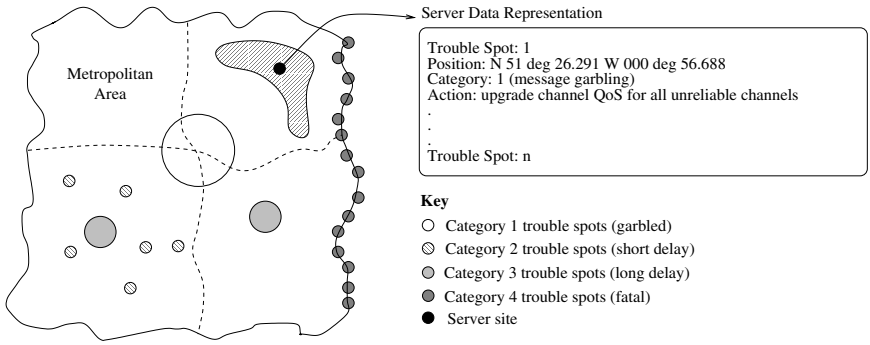


Fig. 2. Geographic Example of Trouble Spot Data

by the occurrence of fault reports through the use of reliable multicast. At this stage, the system must determine (by executing a series of diagnostic tests) which category of trouble spot it has encountered so that it may invoke an appropriate failure handler. In order to achieve this, we propose the following metrics for evaluating the state of a wireless connection in a non ad hoc network.

- *Packet loss* – packet loss at the wireless network interface is a function of not only the power of the received signal, but also the path loss and the state of the network. Packet loss is inversely proportional to signal strength and as such, a high packet loss indicates a drop (or imminent drop) in connection. Thus, packet loss is an important metric in determining trouble spots of the second, third and fourth categories.
- *Path loss* – Wireless network interfaces are designed to operate with a specific signal-to-noise ratio, that is, signal strength to the power of the receiver noise should not fall below a specified value. Note that in free space, signal strength decays inversely with the square of the distance from the source. In a metropolitan environment, this decay is much greater, firstly because of objects and people and secondly, because of destructive interference caused by reflections from these objects. These combine to form path loss which offers a quantitative metric of the signal disruption due to environmental factors. In conjunction with the other metrics, a high path loss indicates a trouble spot of the first category.
- *Response time* – The response time of the network is the duration between a message being transmitted and the reply being received. It can often be useful in classifying trouble spots when the packet loss and path loss are both low. In this circumstance, a high response time will indicate a network problem related to the wired component of the infrastructure.

The above metrics were suggested by a number of sources, the most notable being [8], [23] and the IETF MANET[3] metrics for the evaluation of ad hoc routing protocol performance [3].

[3] Mobile Ad-hoc NETworks working group.

6 Metric Evaluation

For brevity, we only evaluate the performance of the above metrics in relation to trouble spots of the first category, namely, message garbling. More results pertaining to the other categories are available in [22]. The evaluation environment consists of a single laptop connected to the Internet through a Ricochet modem. The laptop is moved along a 10 meter path at a speed of approximately 1 m/s. Message garbling is achieved by obscuring part of the laptops path with several large metal objects. Note that the objective of the experiment is to evaluate the effect of environmental disruptions in terms of the proposed metrics. Thus, an experimental distance of 10 meters is sufficient to achieve the required focus.

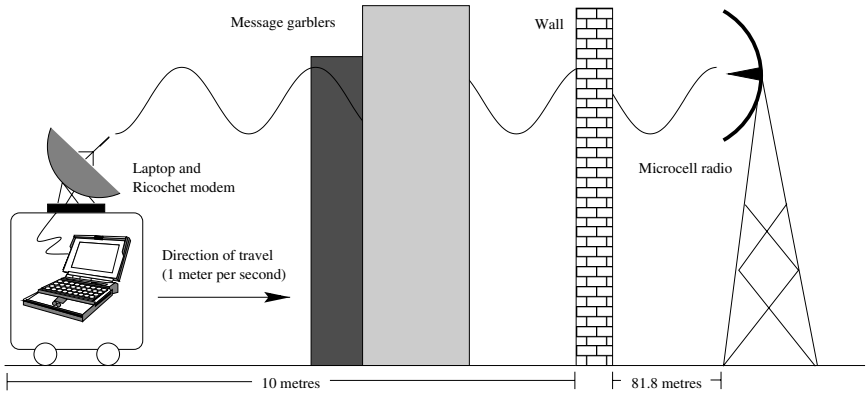


Fig. 3. The Experimental Evaluation Environment

6.1 Packet Loss

Determining a measure of packet loss can be effectively achieved in a number of ways. One such method is to use a reliable transport protocol and to monitor the number of failure reports returned from a fixed number of transmissions. As the CCF implements a reliable multicast primitive, it is this method that we adopt here. Note that as an enhancement, it is often possible to gather packet loss and response time statistics simultaneously.

6.2 Path Loss

The model we use for calculating path loss is the log-distance model as outlined in [20]. There are several models for determining path loss, but a discussion on these is beyond the scope of this paper. Instead, the interested reader is referred to [5, 20]. Using the log-distance path loss model, the average path loss for an arbitrary transmitter receiver (T-R) separation is expressed as a function of distance using the path loss exponent n.

$$\overline{PL}(d) \quad \left(\frac{d}{d_0}\right)^n \; or \; \overline{PL}(dB) = \overline{PL}(d_0) + 10n \, log\left(\frac{d}{d_0}\right) \tag{1}$$

The value n indicates the rate at which path loss increases with distance, d_0 is the close-in reference distance[4] and d is the T-R separation (which in this case varies between 81.8 m and 91.8 m). Table 2 gives the typical path loss exponents for a variety of mobile radio environments.

Environment	Path Loss exponent, n
Free space	2
Urban area cellular radio	2.7 to 3.5
Shadowed urban cellular radio	3 to 5
In building line-of-sight	1.6 to 1.8
Obstructed in building	4 to 6
Obstructed in factories	2 to 3

Table 2. Path loss exponent values for various mobile radio environments

6.3 Response Time

Network response time is typically tested with a simple message transaction. As an example, consider two hosts (denoted by the identifiers A and B). In ascertaining the network response time, host A will record a timestamp before transmitting a message to host B. Host B then immediately returns an acknowledgement message to A who records a second timestamp. The average difference between the two timings is taken as the *round-trip latency* or response time of the network.

6.4 Results

The results from the tests are shown in figure 4 below. Note that the data has been averaged over numerous runs.

The data below show that a network anomaly exists at $\approx$ 87 meters away from the transmitter. The trouble spot is clearly of the message garbling category since both packet loss and response time did not tend towards their maximums (i.e a packet loss of 100% and a response time of would have indicated a disconnection). Due to the relatively small variation in T-R separation, path loss remained approximately constant throughout the experiments. The mean path loss of the experimental environment was calculated, which when compared to the garbled signal strength, indicated that a poor quality (although existent) connection was present. It is envisaged that path loss and signal strength will fulfill a more prominent role when used in city scale testing.

7 Conclusion

In this paper, we have presented a novel approach to fault-tolerance in non ad hoc metropolitan area wireless networks. The discussion began with an overview of the Collaborative Computing Frameworks and gave a technical breakdown of the Ricochet wireless network. We highlighted the differences between ad hoc and non ad hoc wireless networks before postulating that the majority of failures in a non ad hoc metropolitan area wireless network were due to environmental factors. Following this, we proposed a new approach to fault-tolerance which introduced the notion of a trouble spot.

[4] In metropolitan scale cellular systems, it is important to select an appropriate close-in reference distance [14]. In these experiments, we used a value of 200 m.

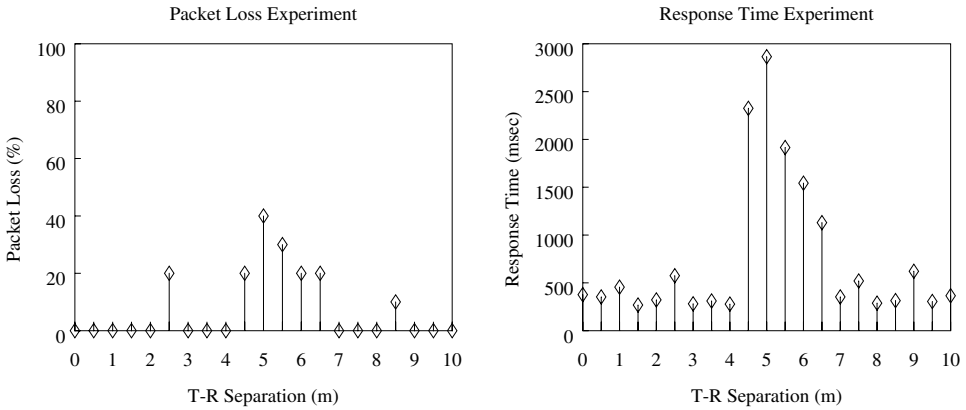


Fig. 4. Response time and packet loss experimental results

We then argue that by categorising trouble spots, the system can take both reactive and pro-active action in eliminating the impact of network failures. This raised the question of how to determine the semantics of a particular trouble spot and to address this, we proposed a set of suitable metrics. In section 6 we experimentally evaluated the effectiveness of these metrics.

This scheme is currently being implemented as part of the CCF in Reading and as such is work-in-progress. When finished, an evaluation and critical comparison will be conducted and reported on in a future presentation.

References

1. E. Bommaiah, M. Liu, A. McAuley, and R. Talpade. *Ad Hoc Multicast Routing Protocol*. IETF Mobile Ad Hoc Networking (MANET) Working Group, August 1998. RFC 2501.
2. M. S. Corson and S. G. Batshell. A Reservation Based Mutlicast Routing Protocol (RBM) for Mobile Networks: Initial Route Construction Phase. *ACM/Baltzer Wireless Networks*, 1(4):427–450, December 1995.
3. S. Corson and J. Macker. *Routing Protocol Performance Issues and Evaluation Considerations*. IETF Mobile Ad Hoc Networking (MANET) Working Group, 1999. RFC 2501.
4. J. J. Garcia-Luna-Aceves and E. L. Madruga. A Multicast Routing Protocol For Ad Hoc Networks. In *Proc. IEEE INFOCOM*, pages 784–792, 1999.
5. J. D. Gibson. *The Mobile Communications Handbook*. CRC Press, 1996.
6. Mobile Ad Hoc Networking (MANET) Working Group. *On-Demand Multicast Routing Protocol (ODMRP) for Ad Hoc Networks*. IETF, 1999. INTERNET-DRAFT.
7. The Bluetooth Special Interest Group. The Official Bluetooth Website, 1999. URL: *http://www.bluetooth.com/*.
8. F. Halsall. *Data Communications, Computer Networks and Open Systems*. Addison-Wesley, fourth edition, 1995.
9. Metricom Incorporated. Ricochet Technology Overview, 1999. URL: *http://www.ricochet.com/ricochet_advantage/tech_overview/*.
10. L. Ji and M. S. Corson. A Lightweight Adaptive Multicast Algorithm. In *Proc. IEEE GLOBECOM*, pages 1036–1042, November 1998.

11. S. Lee, M. Gerla, and C. Chiang. On-Demand Multicast Routing Protocol. In *Proc. IEEE WCNC*, pages 1298–1302, 1999.
12. S. Lee, W. Su, and M. Gerla. Ad hoc Wireless Multicast with Mobility Prediction. In *Proc. IEEE International Conference on Computer Communications and Networks*, pages 4–9, 1999.
13. S. Lee, W. Su, J. Hsu, M. Gerla, and R. Bagrodia. A Performance Comparison Study of Ad Hoc Wireless Multicast Protocols. In *Proc. IEEE INFOCOM*, 2000.
14. W. C. Y. Lee. *Mobile Communications Engineering*. McGraw Hill, second edition, 1997.
15. R. J. Loader and J. S. Pascoe. Future Directions of The CCF Project. Technical report, The University of Reading, Department of Computer Science, 2000. Available by request (in press).
16. R. J. Loader, J. S. Pascoe, and V. S. Sunderam. The Introduction of Fault Tolerance to CCTL. Technical Report RUCS/2000/TR/011/A, The University of Reading, Department of Computer Science, 2000.
17. J. S. Pascoe and R. Loader. A Survey on Safety-Critical Multicast Networking. In *Proc. Safecomp 2000*, October 2000.
18. J. S. Pascoe and R. Loader. The Application of Industrial Multicast Network Technolgies to Robotics. In *Proc. 8th International Symposium on Intelligent Robotic Systems*, July 2000.
19. J. S. Pascoe, R. J. Loader, and V. S. Sunderam. The Implementation and Evaluation of Fault Tolerance in CCTL. Technical Report RUCS/2000/TR/011/A, The University of Reading, Department of Computer Science, 2000.
20. T. Rappaport. *Wireless Communications Principles and Practice*. Prentice Hall, 1996.
21. I. Rhee, S. Cheung, P. Hutto, A. Krantz, and V. Sunderam. Group Communication Support for Distributed Collaboration Systems. In *Proc. Cluster Computing: Networks, Software Tools and Applications*, December 1998.
22. G. Sibley. Ricochet Network Personal Communications. Technical Report 1100-01, Department of Math and Computer Science, 2000. Emory University.
23. Y. Tu, D. Estrin, and S. Gupta. Worst Case Performance Analysis of Wireless Ad Hoc Routing Protocols: Case Study. Technical report, Univeristy of Southern California, 2000.

Tools for Collaboration in Metropolitan Wireless Networks

G. Sibley and V. S. Sunderam

Math & Computer Science
Emory University
Atlanta, GA
30302
{gsibley | vss}@emory.edu

Abstract. This paper presents RRNAPI, a toolkit for accessing network performance in Ricochet Radio Networks. It is envisioned that these tools are a necessary stepping stone in developing frameworks for distributed computing in wireless networks, in particular, extending Collaborative Computing Frameworks (CCF). Many forms of traditional distributed computing require a reliable network, however these programs do not extend to wireless networks because reliable connectivity is not generally possible with wireless networks. This paper explains connectivity issues in Ricochet Networks, presents practical solutions to these problems, and explains a toolkit that can be used when developing distributed applications to address these problems.

Keywords – Wireless, Ricochet, Collaborative Computing

1 Introduction

Today we witness the proliferation of wireless technologies such as IEEE 802.11a/b LAN [10], Bluetooth [6], and Ricochet [11] to name a few. While these technologies and the wireless networks they comprise promise to free users from the desktop, they also pose problems for many network applications. In particular, software that requires consistently reliable networks begins to fail. Many software design techniques used in distributed computing are among those that traditionally require a reliable network. In the context of distributed computing, the terms traditional software or legacy software refers to and includes such projects as Legion [5], GLOBUS [4], HARNESS [2], CCF [9], PVM [16], etc. This whole suite of distributed computing has no inherent mechanism to handle faults of the size and scope that occur in fixed wireless networks. Even in wired networks, it is impossible to achieve 100% reliability. Wireless networks fare much worse; environmental conditions as common as rain or fog can impede communication channels. Faults, or *trouble spots*, last an in-determinant amount of time, typically measured in seconds as opposed to milliseconds. Thus, for legacy software to operate on a wireless network it will need some mechanism for handling trouble spots.

2 Ricochet Radio Networks

Ricochet Radio Networks are micro-cellular based networks in metropolitan areas. Small (shoe box) sized radios are scattered throughout a city atop utility poles at approximately 5 per square mile. These radios have effective coverage areas that overlap creating nearly complete coverage. Environmental features such as buildings and

V.N. Alexandrov et al. (Eds.): ICCS 2001, LNCS 2073, pp. 395–403, 2001.
© Springer-Verlag Berlin Heidelberg 2001

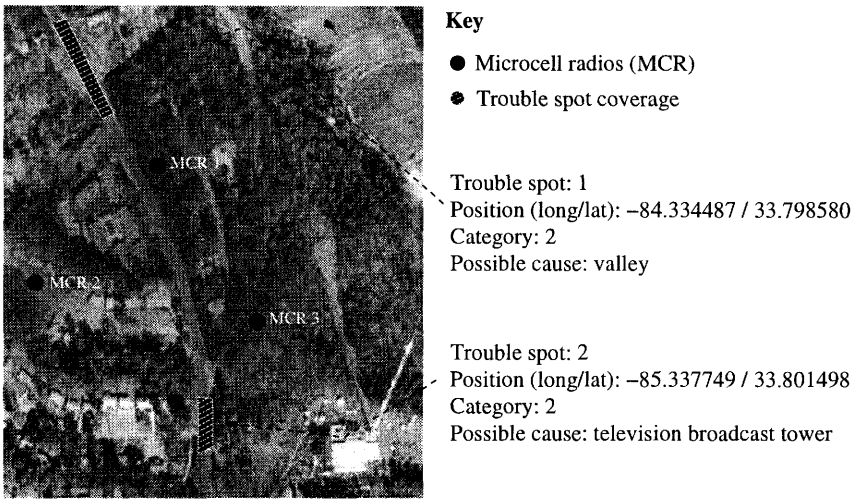


Fig. 1. Typical Ricochet Network and exemplar Trouble spots

bridges will create shadows within which coverage lacks. Further, areas without adequate number of pole-top radios will obviously lack coverage. Poletops relay data to Wireless Access Points (WAPs) which route these packets using a proprietary protocol to a Network Interface Facility (NIF) then to the Network Operations Center (NOC) and finally onto the Internet.

2.1 Trouble Spots

Trouble spots are localized and temporal. That is, a trouble spot exists at a particular position, for a particular amount of time (possibly infinite). (See fig. 1). It is important to note that there are two types of trouble spots; those that are anomalous, and those that result from lack of radio coverage. This later type are not faults per se, rather they are persistent characteristics of fixed wireless networks, while the former are atypical network failures (for example, packet loss due to interference from a passing school bus or from weather conditions.) In section 6 *persistent* trouble spot prediction is discussed.

3 Need for Application Level Awareness

It is desired that loosely connected nodes participating in a distributed computation be tolerant of trouble spots, insofar as trouble spots do not permanently disrupt (or indeed destroy) the distributed application. Ideally, a reliable protocol, such as TCP/IP [7], CCTL [14], or an extension thereof, could handle faults at the transport layer. However, implementing a reliable transport layer on fixed wireless networks may not be a solution. Disconnection times are possibly very long in wireless networks [15], so while the transport layer may still be working, the application above it may not be. However, there is significant information available to an application concerning connection quality. Thus, a useful protocol would handle indeterminantly long disconnections, as well as be aware of the quality of service provided by the network. These issues are of paramount interest to an application that runs *on top* of the transport layer.

Todays wireless networks are inherently faulty to such a degree that transport protocols alone cannot adequately address problems that arise in distributed applications. Therefore, distributed applications must act intelligently by managing network usage with regard to the networks current state. Our experience [11] with Ricochet Networks suggests that as much as 20% of the time a modem is in service is spent within trouble spots that last longer than one second. While most transport layer protocols can handle faults of this nature, they do not offer the application the ability to easily assess the situation (i.e. quantify such metrics as proposed in [12] and discussed in section 3.1. Thus, if necessary, (as is in Collaborative Computing Frameworks [1]), protocols for network control should be embedded within the application layer. To be useful, these protocols should offer an application knowledge about network connectivity. Next we examine how to determine quality of connection.

3.1 What Metrics Are Useful?

From our previous work in evaluating Ricochet Radio Networks [12] three metrics have been selected as useful when determining quality of connection. These are latency, packet loss, and signal strength. Latency and packet loss can each be calculated using datagram sockets and simple echo requests. The time period over which these statistics are generated should of course be variable to accommodate different applications. Signal strength is a quantity measured in decibels that in Ricochet Radio Networks typically ranges from -30 dB to -110dB. This information is available via queries sent to Ricochet modems AT commands. A modem gathers this data from all poletops it can detect.

3.2 Other Useful Information

Poletops provide other information [8] beyond signal strength such as their latitude and longitude, and their color. Among other things, color specifies whether or not a radio is a WAP (color is not an immediately useful property). Latitude and longitude can be used to give rough estimates of user position. The more radios are visible, the more accurate that estimation becomes. Temporal positional data can also be used to make estimates of the users heading. One can see how positional information is useful for connectivity predication as it relates to application level awareness.

We have discussed two general types of trouble spots in Ricochet Radio Networks, how to assess the severity of a trouble spot, and why it is necessary to offer this information in an application level interface. Next we look at RRNAPI, a toolkit to serve this task.

4 Ricochet Radio Network Application Program Interface

The Ricochet Radio Network Application Program Interface was developed in order to enable programs that wish to take intelligent action based on network performance. The API offers 21 functions (see appendix A). Programs instantiate RRNAPI by calling `RRN_init()`. This function must be called before any other RRNAPI function; `RRN_done()` is called when the application is finished with the API. The time from initialization to completion constitutes a 'session'. Depending on how the `mode` parameter is set the API spawns a thread that handles IO from the modem, the upper level API, the RRNAPI Server, and session files. This thread updates the client application

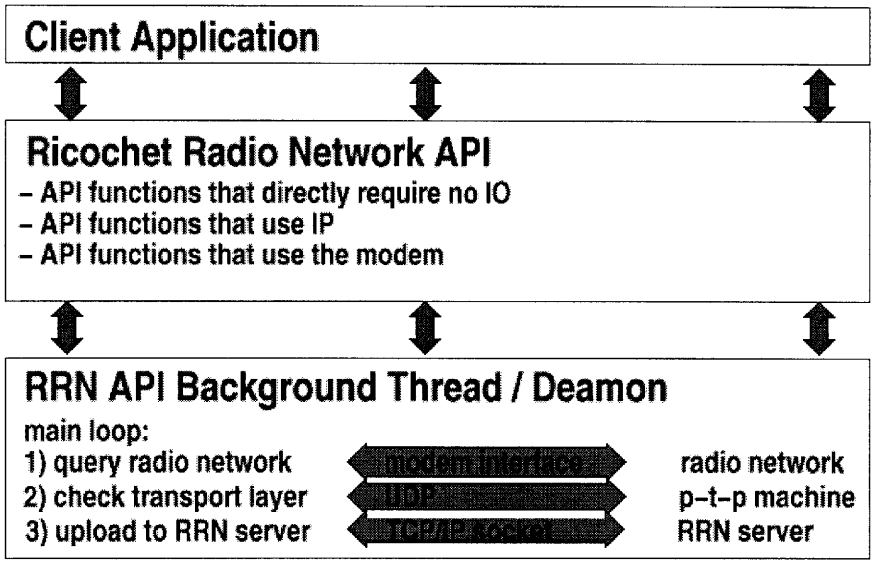


Fig. 2. Functional diagram for RRNAPI

at programmer specified intervals with 'snapshots' (see appendix B for a list of important data structures) of the current network statistics. These snapshots can be saved and loaded to and from disk or sent as updates to the RRNAPI Server.

Applications can choose the update threads behavior via the mode parameter to `RRN_init`. These flags (see fig. 3) can either update the server or not (`RRN_DO_UPDATE`), use the modem to query the network or get this information from a file (`RRN_USE_MODEM` - this is useful for developing the set of internal functions that communicate with the modem directly), continuously cache the session to disk (`RRN_RECORD`), attach to a modem that is online or off-line (`RRN_MODEM_MASTER`), and specify whether or not to perform the transport layer check using UDP (`RRN_CHECK_TL` - turning this off will render useless further calls to `RRN_packet_loss` and `RRN_latency`). The default initialization mode for `RRN_init` will update the RRN Server, open the modem off line, check the transport layer, and cache the session.

`RRN_CHECK_TL` enables the `rrn_check_udp_thread`. This thread uses ICMP echo request packets to estimate latency and packet loss. The time period over which these calculations are performed and the interval between checks is set by `RRN_config_tl_check`. This function also allows the application to set a latency threshold that specifies a maximum time period after which latency is considered to be infinity. This is useful for programs that require performance within set parameters. The default values are 10 ICMP packets over 10 seconds with a an infinite (`INT_MAX`) threshold. Error codes are set in the global rrn_error. The values of errors can be seen in figure 3. For a complete listing of RRNAPI see Appendix A.

5 Technical Notes

RRNAPI works only with Ricochet Radio Networks, and is therefore useful to only a small niche of wireless applications. While this may seem like a drawback at first, it is

```
RRN Error Codes:

#define RRN_CONNECT        1
#define RRN_NO_DIAL_TONE   (2   | RRN_ERROR)
#define RRN_NO_ANSWER      (4   | RRN_ERROR)
#define RRN_NO_CARRIER     (8   | RRN_ERROR)
#define RRN_RING           16
#define RRN_BUSY           (32  | RRN_ERROR)
#define RRN_IO_ERROR       (64  | RRN_ERROR)
#define RRN_INIT_ERROR     (128 | RRN_ERROR)
#define RRN_MODEM_ERROR    (256 | RRN_ERROR)
#define RRN_TIMEOUT_ERROR (512 | RRN_ERROR)

RRN Location Modes:

#define RNN_RSSI_MID       2
#define RNN_RSSI_NEAR      1
#define RNN_RSSI_AVG       0

RRN Initilization Modes:

#define RRN_DO_UPDATE      1
#define RRN_USE_MODEM      4
#define RRN_USE_STDIO      8
#define RRN_RECORD         16
#define RRN_MODEM_MASTER   32
#define RRN_MODEM_SLAVE    64
#define RRN_CHECK_TL       128
```

Fig. 3. RRNAPI Error Codes.

important to note that the API's usefulness stems from the ability to do three things: 1) evaluate transport layer performance, 2) measure signal strength, and 3) estimate client location. Both one and two above are provided by most wireless devices (for example, Sprint PCS [13], 802.11b [10] compliant cards, and CDPD [3]). However, point three is not provided nor accessible in most wireless devices. This is the only limitation to extending RRNAPI to a more general wireless device API. Gathering location information is trivial and accurate with the use of GPS.

The use of different devices, and indeed different modems of the same make, results in different data sets for the *same* areas in a network. Thus, data gathered from Metricom GS modems is not equivalent to data from Sierra Wireless modems or Novatel modems; nor are results from one Metricom GS the same as those of another Metricom GS. While the ability to generally model network performance is important, there is a need to examine these models in light of which hardware is in use.

5.1 Extending RRNAPI

If GPS devices become pervasive in mobile electronics, then extending RRNAPI to work on a broad range of devices is possible. One should note that the fore mentioned issue of differentiating between types of modem is more imperative when the difference is not between modems, but between entire technologies (e.g. the difference between wireless LAN and Ricochet are vast when compared to the differences between one make of Ricochet card and another). Thus, for a wireless API to be useful (in the sense that RRNAPI is useful) it must include location information.

RRNAPI is designed with a modular back end to support different modem hardware. Modem specific functions in RRNAPI are defined separately. Anyone wishing to implement RRNAPI with a modem other than the Ricochet GS serial modem will have to rewrite the functions in `modem.c` and compile it with `rrnapi.c`. Later, these internal functions will be dynamically loadable shared objects. Further, it may be useful to have the API attempt to detect the modem make of an interface and then load an appropriate function set.

Currently, RRNAPI requires Glib, a POSIX threads implementation, a Linux host, and a working Ricochet GS Serial Modem. Porting RRNAPI to other Unix platforms should be straightforward. Porting to Microsoft Windows will require some effort.

6 Future Work

The RRNAPI offers not only an interface for mobile clients to access network performance, but also the ability to query a remote database that contains past network performance. Ricochet Radio Network Daemons and Ricochet Radio Network Terminate Stay Resident (RRND's and RRNTSR's) could be run on client computers that continually update an RRN Server with network statistics. This server holds temporal data on the networks performance from the clients point of view. Given enough data it should be possible to create an accurate picture of network performance at some time in the future. This allows for the prediction of trouble spots. All programs that use the RRNAPI get their network data (packet loss, signal strength, etc.) via a thread that queries the modem and network at application specified intervals. There is also the option to have this thread send network performance data to the RRN Server. Such a database opens many possibilities for research. What techniques would be used to model performance? What accuracies are possible? Is it possible to predict temporal performance of the network? Can network load be detected? These questions require further investigation.

A Appendix: RRNAPI Listing

The RRNAPI is available from `http://vector.mathcs.emory.edu/rrnapi/`

`RRN_init()`	Input: modem device name, mode bits. Output: returns zero on success. Sets `rrn_error` otherwise. Must be called before the API is used. Connects to the modem and starts background IO threads.
`RRN_done(void)`	Input: void. Output: return zero on success. Sets `rrn_error` otherwise. Called to close the API.
`RRN_set_rrn_file()`	Input: string name of file Output: void Set file for session.
`RRN_load_rrn_file()`	Input: pointer to a linked list of snapshots. Output: same list. Loads previous session from file into a linked list of snapshots.

`RRN_save_rrn_file()`	Input: pointer to a linked list of snapshots. Output: same list. Saves session into file of snapshots.
`RRN_poletop_count()`	Input: a list of poletops. Output: how many poletops are on the list. If input is null, returns the number of poletops currently visible to the modem.
`RRN_rssi()`	Input: poletop list, calculation mode. Output: a radio signal strength indication (rssi). Uses a list of poletops to calculate rssi. `mode` specifies how to calculate the rssi; use `RRN_RSSI_AVG` for an average of all poletops, or `RNN_RSSI_NEAR` to get the strength of the closest (actually, this is the strongest, not necessarily the closest) poletop.
`RRN_connection_status()`	Input: void. Output: an error code.
`RRN_network_id()`	Input: void. Output: Ricochet Network Number.
`RRN_firmware_version()`	Input: void. Output: string. Returns the modems software version.
`RRN_hardware_version()`	Input: void. Output: string. Returns the modems hardware version.
`RRN_poletops()`	Input: void. Output: linked list of `RNN_poletop_t`'s. Returns a list of poletops sorted by location that are 'visible'.
`RRN_print_poletops()`	Input: list of `RRN_poletop_t`'s. Output: prints to stdout. Prints a list of `RRN_poletop_t`'s.
`RRN_set_update_timeout()`	Input: a time in milliseconds. Output: void. Set how often to query the modem, check the transport layer, and update the server.
`RRN_packet_loss()`	Input: void. Output: a percentage. See what percentage of ICMP echo request packets were lost over a set time period.
`RRN_latency(void)`	Input: void. Output: time in milliseconds. Check the average latency over a set time period.
`RRN_set_heading()`	Input: a `RRN_vector_t` pointer. Output: void. RRNAPI maintains a heading. This is a vector from the last location known to the current location with a velocity magnitude.

`RRN_location()`	Input: a mode. Output: a lat. long. location. Calculate likely user location based on radio locations and signal strength. mode 0 = weighted average of active poletops. (default). mode 1 = strongest, and most likely closest, poletop. mode 2 = center without respect to signal strength. These modes are `RRN_RSSI_AVG`, `RRN_RSSI_NEAR` and `RRN_RSSI_MID`.
`RRN_print_modem()`	Input: a RRN_modem_t. Output: void. Print modem information.
`RRN_modem_info()`	Input: void. Output: pointer to a RRN_modem_t. Get modem information. This does NOT query the modem, it just sees what `RRN_init` found when the API was started.
`RRN_config_tl_check()`	Input: number of ICMP packets to send, interval between packets, threshold for latency. Output: void. Establishes the behavior of `rrn_check_udp_thread`.

B Appendix: Data Structures

```
typedef struct _vector_t{
  double base_long;
  double base_lat;
  double mag_long;
  double mag_lat;
}RRN_vector_t;

typedef struct _poletop_t{
  double latitude;
  double longitude;
  int    strength;
  int    color;

}RRN_poletop_t;

typedef struct _udpinfo_t{
  unsigned int latency;
  unsigned int latency_threshold;
  float        packet_loss;
  unsigned int delta;
  unsigned int width;
} RRN_udpinfo_t;
```

```
typedef struct _snapshot_t{
  GList                 *poletop_list;
  struct timeval        *timestamp;
  RRN_udpinfo_t         *tl;
  RRN_vector_t          *heading;
} RRN_snapshot_t;
```

References

1. S. Chodrow, S. Cheung, P. Hutto, A. Krantz, P. Gray, T. Goddard, I. Rhee, and V. Sunderam. CCF: A Collaborative computing frameworks. In *IEEE Internet Computing*, January / February 2000.
2. J. Dongarra, A. Geist, J. Kohl, P. Papadopoulos, and V. Sunderam. Harness: Heterogeneous adaptable reconfigurable networked systems. In *High Performance Distributed Computing*, 1998.
3. Wireless Data Forum. Cdpd system specification release 1.1, 1998. URL: *http://www.wirelessdata.org/develop/cdpdspec/index.asp.*
4. I. Foster and C. Kesselman. Globus: A metacomputing infrastructure toolkit. *International Journal of Supercomputer Applications*, 1997. URL: *http://www.globus.org/research/papers.html.*
5. A. S. Grimshaw, W. A. Wulf, J. C. French, A. C. Weaver, and P. F. Reynolds Jr. A synopsis of the legion project. Technical Report CS-94-20, Department of Computer Science, University of Virginia, June 1996. URL: *http://www.cs.virginia.edu/ legion/CS-94-20.pdf.*
6. The Bluetooth Special Interest Group. The Official Bluetooth Website, 1999. URL: *http://www.bluetooth.com/developer/whitepaper/whitepaper.asp.*
7. E. A. Hall. *The Core Internet Protocols: The Definitive Gude.* O'Reilly & Associates, 2000.
8. Metricom Incorporated. Ricochet Technology Overview, 1999. URL: *http://www.ricochet.com/ricochet_advantage/tech_overview/.*
9. R. J. Loader and J. S. Pascoe. Future Directions of The CCF Project. Technical report, The University of Reading, Department of Computer Science, 2000. Available by request (in press).
10. IEEE White Paper. Ieee 802.11b standard. Web Page, 2000. URL: *http://www.wlana.com/learn/80211.htm.*
11. J. S. Pascoe, G. Sibley, V. S. Sunderam, and R. J. Loader. Mobile Wide Area Wireless Fault Tolerance. Technical report, University of Reading and Emory University, 2001.
12. J. S. Pascoe, G. Sibley, V. S. Sunderam, and R. J. Loader. Mobile Wide Area Wireless Fault Tolerance. Technical report, University of Reading and Emory University, 2001.
13. Sprint PCS. Sprint pcs developers forum, 2000. URL: *http://www.developer.sprintpcs.com/.*
14. I. Rhee, S. Cheung, P. Hutto, A. Krantz, and V. Sunderam. Group Communication Support for Distributed Collaboration Systems. In *Proc. Cluster Computing: Networks, Software Tools and Applications*, December 1998.
15. G. Sibley. Ricochet Network Personal Communications. Technical Report 1100-01, Department of Math and Computer Science, 2000. Emory University.
16. V. S. Sunderam. Pvm: A framework fo parallel distributed computing. *Concurrency, Practice and Experience*, December 1990.

A Repository System with Secure File Access for Collaborative Environments ⋆

Paul A. Gray†, Srividya Chandramohan‡ and Vaidy S. Sunderam‡

†Dept. of Computer Science
University of Northern Iowa
Cedar Falls, Iowa
50614-0506
gray@cns.uni.edu

‡Dept of Math & Computer Science
Emory University
Atlanta, Georgia
30332
{schand2 | vss}@emory.edu

Abstract. Collaborative computing environments which allow remote execution of applications, need a remote storage facility that supports shared access to the software resources required for computation. In this context, there is also a need to guarantee authorized and secure access to the shared resources. This paper investigates the use of a repository system in collaborative computing environments and discusses techniques to provide privacy, user authentication and access control to the repository using certificates. A protocol based on SSL is developed for query processing. The IceT environment is used as an exemplar for the application of the secure repository system.

1 Introduction

In recent years, the use of network environments as a platform for high performance distributed computing, has become popular. Research collaborations are being formed by merging geographically-distributed environments [5]. These collaborations often pool resources together in order to tackle a common goal.

Each member in any of the groups can contribute software and data to be shared by other members of the group. There are several means to share data across networks varying from simple file transfer to complex distributed database management and distributed shared memory approach. These methods for file sharing lack the ability to support a dynamically changing membership of users within the groups. They also lack mechanisms to authenticate users in dynamic environments.

One model is to have applications and data stores in a repository to facilitate group access. Due to the dynamic nature of the collaborative resource alliance, the repository system cannot be managed with physical user accounts on the machines that host the repositories. In this paper, we present a simple model of a repository system that suits the computing requirements of dynamic collaborative environments. Section 2.2 outlines the design and discusses the usefulness of this repository system.

When an alliance is formed and data from the repository has to be shared, issues such as secure communication, authentication and authorization have to be tackled. The Grid Security Infrastructure (GSI) [9], developed within the Globus project [8] addresses these issues in detail and defines a security policy by mapping interdomain

⋆ Research supported by NSF grant ACI-9872167 and the University of Northern Iowa's Graduate College.

V.N. Alexandrov et al. (Eds.): ICCS 2001, LNCS 2073, pp. 404–412, 2001.
Springer-Verlag Berlin Heidelberg 2001

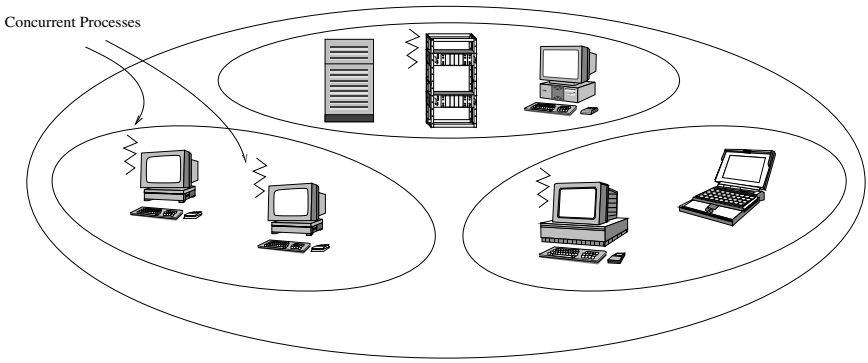


Fig. 1. Dynamic Collaborative Environment.

operations to local security policy. Further, Akenti [16] implements an automated access control mechanism using digitally signed certificates.

Lack of a local security policy or local database administration and the dynamic nature of the collaboration necessitate development of an access control mechanism that is more dynamic and adaptable than the GSI and Akenti infrastructures. The relevant security policies and associated protocols for query processing used in this model are described in Section 3. We also present a reference implementation in the context of IceT specifically [17], and applicable to the repository frameworks in general.

2 The Repository System

In this section we discuss the background and need for a repository system. The details of the proposed model are also outlined.

2.1 Background and Need

The Internet enables users to access resources and run applications over a heterogeneous collection of computers and networks. Figure 1 shows an example of a dynamic environment in which three heterogeneous, networked environments have been merged to form a single virtual machine. Processes can now migrate across environments for optimal resource utilization.

In these and other scenarios, when code is sent from one computer to another and run at the destination, problems such as portability of machine code, commanality of data representation formats and data conversion for network transfer arise. For example, PC users sometimes send executable files as email attachments to be run by the recipient, but a recipient will not be able to run it, for example, on a Macintosh computer. These problems are overcome using 1) the *virtual machine* approach, such as in Java, as a way of making code executable on any hardware, and 2) *External data representation* as an agreed standard for the representation of data. Sun XDR, CORBA's common data representation (CDR) and Java's object serialization are examples of the latter.

Given the use of a suitable scheme to mask the heterogeneity in distributed systems, this paper looks at some specific computing requirements that are not effectively met by current systems and provides a solution for the same.

Consider the case where Java programs are used in a collaborative environment. These programs may access legacy codes written in C, C++ or Fortran for high performance computations. Let Bob and Alice be two end communicating parties (or machines). Bob sends program called "foo" to Alice via a communication link. To run "foo", Alice has to: locate and resolve static dependencies (java classes and methods), locate and link external shared libraries (e.g. a DLL for a windows machine) and provide for external user data that the program might require for processing.

These dependencies can be easily resolved if Alice can locate the files on her local filesystem. But Alice has no prior knowledge of the resources required and need not maintain a reserve. Alice could interactively request Bob to send these files and Bob can respond back with the required files. If Bob sends the program to Sue as well, with Sue working on another machine within the same network as that of Alice, Bob has to duplicate the file transfer. Suppose that Bob wants to do some statistical analysis and needs Carol and Dave on a different network to participate in the computation concurrently, Bob has to maintain persistent connection with all four parties and service their needs. This will be a potential performance bottleneck. Alternately, in an agent-based scenario, Bob could generate an agent process on Alice's machine. Then Bob's machine is not needed again even if the process migrates to Carol's system.

Irrespective of the programming paradigm used, such problems persist in collaborative computing environments. *There is no Network File System for such environments* that facilitates shared access to applications, libraries and user data for all members in the communion, catering to the different underlying operating systems and architectures. In this vein, we propose a simple *repository system* for dynamic environments, supporting mobile agents (by serving up files tailored to operating system/architecture) and remote data storage. Continuing with the example scenario, Bob can now store the Java files with architecture and operating system-specific libraries in a repository and authorize Alice, Sue, Carol and Dave to access the repository. When the group is formed, the parties are given the knowledge of the available repositories so that they can fetch data on-demand.

2.2 Details of the Model

The scope of the repository system is to address the computing needs of dynamic environments in which processes participate in remote execution and remote access of resources within the virtual machine environment. Java applications that use legacy codes for high performance computations are targetted. The repository system fills in the need for a shared filesystem. In this model, which epitomizes the breadth of the implementation, users store Java-based front-ends and supporting native library formats for different architectures that might be called upon to run processes. Figure 2 shows possible contents of a repository.

The model also supports the following features:

- Repositories are not restricted to locations within the virtual machine. Bob can create a repository on his machine and so can Alice. Bob and Alice could be within the same intranet or otherwise.
- Users can add to or delete from, a local or remote repository based on a global access policy.
- The owner of a resource can impose security restrictions for availability and accessibility of the resource. There is no central authority to impose access restrictions on all the resources.

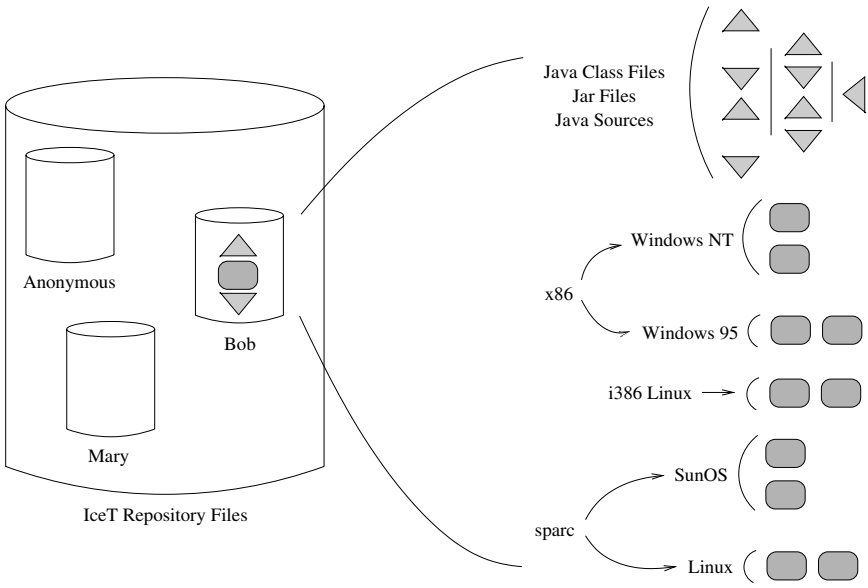


Fig. 2. Repository Files Belonging to Users.

The dynamic nature of the collaboration and the lack of central authorities, hinder the application of conventional distributed database management models where database access is administered by setting up user accounts and passwords. There is a need to specify access control policies that are non-account based and do not rely upon the presence of a database administrator. The access control policy for the repository system is described in Section 3.

We adopt a User Interface Distribution model (see [12]) in the three-tier client/server architecture. The layers are represented by *hosts*, *clients* and *servers*. Figure 3 illustrates the model. Host performs data-access processing i.e accesses data from the disk. Client performs user interface processing. It contains GUI interfaces and additional rules such as client certificates (for authentication). Server performs function processing. It stores constraints that are used to access data from the host.

Instead of accessing the repository via a standard interface (such as JDBC or ODBC), the client sends queries to the server and the server processes the requests. Thus the server acts as a conduit for passing processed data from host to client. The viablity of this approach stems from two reasons: (1) the client can run in a computing system different from the host and the server, and (2) access control mechanisms can be implemented independent of the type of interface to the repository system.

3 Security, Authorization and Access Control

The centralized storage of data in databases, and the accessing of this data by multiple end users, bring with them a need for security. The model must have mechanisms that will allow users to access data they need yet prevent them from accessing data they are not authorized to see. In addition to controlling the data a particular user has access to,

the repository system should control the type of access the user has, such as whether the user is allowed only to retrieve the data or may also make changes to it or add new data to the database.

As noted in Section 2.2, due to the dynamic characteristics of the alliance, where new users can join and existing users can leave the group at anytime, users do not have accounts and passwords on the repository host. Hence user authentication mechanisms which depend on account setup such as Kerberos [13] and SSH [11] cannot be used. This also eliminates risks due to "password sniffing" [7].

Instead, Certificates are used both for authorizing access to the repository and for governing subsequent access to files. The group has to maintain Certificate Authorities (CA) which issue and sign certificates for the group members. The repository server stores this "trust" information. When Bob inserts his files into the repository, he presents to the repository server, a valid certificate digitally signed by a CA that the server trusts. This authorizes Bob to access the repository. Bob could grant access privileges to Alice by signing Alice's certificate already signed by a group CA. This creates a certificate chain of users. Along with the files owned by Bob, access permissions in the form of user certificates will be stored in the repository. Bob could also specify some of his files to be public so that anonymous users can read those files. The server has to check the permissions before granting access. When Bob changes his mind, and wants to revoke privileges granted for Alice, he will revoke Alice's certificate. A Certificate Revocation List (CRL) is maintained by the group authority for each user. Alternately, the resource alliance can be terminated and all access privileges are revoked for all users. In both cases, *a consistent state of the repository on the host has to be maintained*. The access control list for a user in the repository has a "checkCRL" flag. Revocation of privileges is reflected by setting the flag, so that the server can query the user's CRL prior to servicing any file requests by the user.

When a user posts requests, the repository server's reply may constitute a file transfer. Data is typically sent via an insecure communication channel. To ensure privacy , the data has to be encrypted. This issue has been addressed in depth in projects such

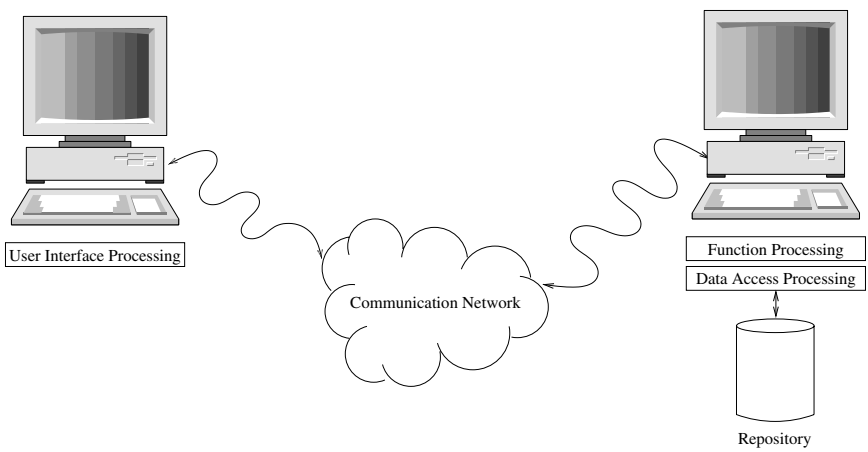


Fig. 3. User Interface Distribution Model.

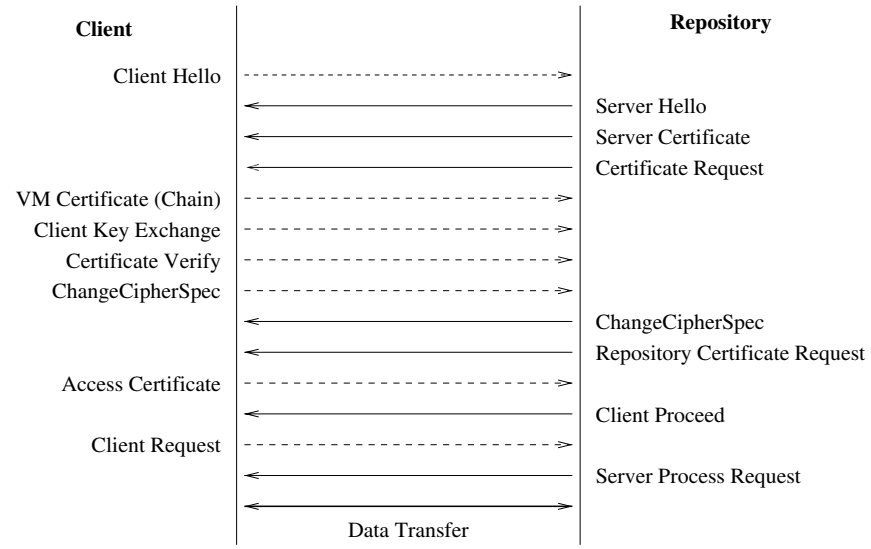


Fig. 4. The Protocol Established For Accessing Files In A Repository.

as Globus [8], Akenti [16] and securePVM [18]. The channel should also be protected from network attacks such as eavesdropping, masquerading, message tampering, replaying, denial of service etc. (see [14]). This calls for making the channel secure. The SSL protocol [4] can be used to establish a secure channel. In an open network, all client parties may not use the same client software. The client and server software may not support same encryption algorithms. SSL is a good choice because it is designed so that the algorithms used for encryption and authentication are negotiated between the processes at the two ends of the connection during the initial handshake.

3.1 Secure Query Processing Protocol

The protocol used to establish contact with the repository is similar to SSL, with modifications to allow the use of certificate chains belonging to users in the merged environments. Figure 4 depicts the protocol.

The protocol messages are sent in the following order:

1. Client hello - The client sends the repository server information including the highest version of SSL it supports and a list of the cipher suites it supports. The cipher suite information includes cryptographic algorithms and key sizes.
2. Server hello - The server chooses the highest version of SSL and the best cipher suite that both the client and server support and sends this information to the client.
3. Server Certificate - The repository server sends the client a certificate or a certificate chain. This message is used to authenticate the repository server.
4. Certificate request - The server then issues this message to the client, which contains a list of acceptable Distinguished Names (DN) that are recognized as credible.
5. VM Certificate (Chain)- The client sends its certificate (chain), just as the server did in Message 3. The client has to send a certificate that has been certified by one of the listed DNs.

6. Client key exchange - The client generates information used to create a key to use for symmetric encryption.
7. Certificate verify - The client sends information that it digitally signs using a cryptographic hash function. When the server decrypts this information with the client's public key, the server is able to authenticate the client.
8. Change cipher spec - The client sends a message telling the server to change to encrypted mode.
9. Change cipher spec - If the server accepts the certificate as valid, a response to the change in cipher is issued.
10. Repository Certificate Request - At this point, the client has authenticated itself to the server and a secure channel has been established. However, another valid certificate is required to access files on the repository as the owner of the files may have imposed access restrictions. The server requests the client for the certificate corresponding to the owner of the repository files.
11. Access Certificate - The client's certificate signed by the owner is presented if one is available, else a NoCertificateAlert message is sent.
12. Client Proceed - If the server can verify the certificate presented, it grants access to the client. Otherwise the access is denied and the session is closed. It allows anonymous access if it received a NoCertificateAlert in message 11.
13. Client Request - The client requests to access or update files in the repository and sends the details (file name, architecture, OS) of the file it requests.
14. Data Transfer - The server queries the access control list to check if the particular file can be accessed. If the checkCRL flag is set, the client's CRL is checked for validity before granting access to the file.

4 A Reference Implementation of the Repository System

In this section, we describe the IceT repository system, an implementation of our proposed model. The repository system was developed as a part of IceT project [17], whose focus is to:

- build distributed applications using multiple heterogeneous environments.
- support the use of portable shared libraries.
- target applications to dynamic reconfigurable environment which allows merging and splitting of environments.
- address security concerns
- provide an environment suitable for collaboration and distributed computing applications

We describe here selected aspects of this implementation, focusing on our use of the Java Secure Socket Extension (JSSE) API for SSL, a Postgres database and Java Keytool for certificates.

4.1 Use of Java Secure Socket Extension API and Keytool

The Java Secure Socket Extension (JSSE) [1] provides a framework and a reference implementation for a Java version of the SSL protocol and includes functionality for data encryption, server authentication, message integrity, and client authentication. The JSSE API is used for creating and configuring secure socket factories. The Java Keytool is used to generate keys (inserted into a keystore), certificate signing requests and

to import trusted X509 certificates into a user defined truststore. For details of using x.509 certificates see [10]. For testing purposes, OpenSSL `ca` command (as in [2]) was used to sign the certificate requests and create a chain of trusted of X509 certificates. The keystore and truststore are loaded into X509 key and trust managers respectively. The API also provides a class representing a secure socket protocol implementation. A Query Processing protocol as described in Section 3.1 is implemented on top of SSL to access the repository. These tools come together in our implementation to facilitate dynamic, short-term collaborative alliances,where self-signed certificates and CRLs are used on a more intimate framework of users. I.e. users will hold self-signed certificates and act as their own certificate authorities.

4.2 Postgres Database

PostgreSQL [3] was our implementation choice for the database. Postgres uses a simple "process per-user" client/server model. A Postgres session consists of a supervisory daemon process, the user's frontend application (eg. psql program) and one or more backend database servers. A single postmaster manages the repositories on the host. The postmaster is always running, waiting for requests, whereas frontend and backend processes come and go.

A primary registry table is created to store X509 certificates of different users and their corresponding *user-id*. A table indexed by *file-id* stores Java source files for each user and a library table stores the user's libraries based on architecture and operating system. A separate access permissions table maintains access list for files belonging to a user. Any query posted to the repository has to be processed as:

$$\text{user certificate} \rightarrow \text{user-id} \rightarrow \text{file-id} \rightarrow \text{file permissions} \rightarrow \text{file contents}$$

I.e., each access to the repository requires a valid X.509 certificate and access permissions are checked prior to granting file access.

Once a secure channel is established using JSSE methods, the user sends his queries to the server. The server connects to the postgres host (on same machine as the server) via the JDBC interface. The user certificate is matched to that in the repository, access permissions are checked and the encrypted file is transferred to the requesting client.

5 Conclusions and Future Work

We have described a model for distributed access of resources with security and authorization features. The model is general enough to be suited to several applications including but not limited to:

- collaborative computing projects such as Harness [15] and CCF [6]
- security policy for remote execution and remote access of resources
- extending security architecture given by Globus and Akenti for dynamic environments
- access control policy for distributed databases using certificates

The prototype implementation has shown the feasibility of the certificate-based authentication and secure repository model. Preliminary benchmarks have also shown that there can be considerable overhead associated with SSL channel initialization and encryption of the data stream. Our current design does not clearly specify interactions

between the repository system and the distributed applications in which it is used. It does not yet address functions pertinent to distributed databases such as transparency control, concurrency etc. Some of the improvements in the security policy include providing for validation of files within the repository using message digests and encrypting data stored in the repository. Our future work also includes implementing the repository system using a language-independent scheme so that it can be easily adapted to existing systems.

References

1. URL: http://java.sun.com/products/jsse.
2. URL: http://www.openssl.org/doc/.apps/openssl.html.
3. PostgreSQL user's guide. URL: http://www.postgresql.org/docs/user/user.html.
4. A. O. Freier, P. Karlton and P. C. Kosher. *The SSL Protocol, version 3.0.* Netscape Communications, Internet Draft, Nov 1996. URL: http://www.netscape.com/eng/ssl3/.
5. C. Catlett and L. Smarr. Metacomputing. *Communications of The ACM*, 35(6):44–52, 1992.
6. S. Chodrow, S. Cheung, P. Hutto, A. Krantz, P. Gray, T. Goddard, I. Rhee, and V. Sunderam. CCF: A Collaborative Computing Frameworks. In *IEEE Internet Computing*, Jan/Feb 2000.
7. Computer Emergency Response Team. Ongoing Network Monitoring Attacks. CERT Advisory: CA - 94:01, Feb 1994.
8. I. Foster and C. Kesselman. Globus: A Metacomputing Infrastructure Toolkit. *International Journal of Supercomputing Applications*, May 1997.
9. I. Foster, C. Kesselman, G. Tsudik, and S. Tuecke. A security architecture for computational grids. In *ACM Conference on Computers and Security*, pages 83–91. ACM Press, 1998.
10. Internation Telecommunication Union. X.509: Information Technology - open systems interconnection - the directory: Public-key and attribute certificate frameworks. ITU-T Recommendation, Mar 2000. To be published.
11. J. Barrett and R. Silverman. *SSH, The Secure Shell: The Definitive Guide*. O'Reilly, 1st edition, 2001.
12. J. Martin and J. Leben. *Client/Server Databases - Enterprise Computing.* Prentice Hall P T R, 1995.
13. J. Steiner and C. Neuman and J. Schiller. Kerberos: An Authentication Service for Open Network Systems. In *Usenix Conference Proceedings*, 1988.
14. M. Dekker. Security of the Internet. *The Froehlich/Kent Encyclopedia of Telecommunications*, 15:231–255, 1997. URL: www.cert.org/encyc_article/tocencyc.html.
15. M. Migliardi and V. Sunderam. Heterogeneous Distributed Virtual Machines in the Harness Metacomputing Framework.
16. M. Thomson, W. Johnston, S. Mudumbai, G. Hoo, K. Jackson, and A. Essari. Certificate-based access control for widely distributed resources. In *Proceedings of the Eighth Usenix Security Symposium*, Aug 99.
17. P. Gray and V. Sunderam. IceT: Distributed Computing and Java. *Concurrency: Practice and Experience*, 11(9):1161–1167, Nov 1997.
18. T. H. Dunigan and N. Venugopal. Secure PVM. Technical Report TM-13203, Oak Ridge National Laboratories, Aug 1996.

Authentication Service Model Supporting Multiple Domains in Distributed Computing

Kyung-Ah Chang, Byung-Rae Lee, Tai-Yun Kim

Dept. of Computer Science & Engineering, Korea University,
1, 5-ga, Anam-dong, Sungbuk-ku, Seoul, 136-701, Korea
{gypsy93, brlee, tykim}@netlab.korea.ac.kr

Abstract. In this paper, based on CORBA security service specification[1, 3], we propose the authentication service model supporting multiple domains for distributed computing with an extension to the Kerberos[13] authentication framework using public key cryptosystem[15]. This proposed model supports the protection of the high-level resources and the preservation of the security policies of the underlying resources that form the foundation of various domains, between the Kerberized domains[14] and the Non-Kerberized domains. Also we achieved the flexibility of key management and reliable session key generation between the Client and the Provider using the public key cryptosystem.

1 Introduction

The traditional requirements of security mechanisms and policies are exacerbated in the current distributed computing, as the physical resources of this exist in multiple administrative domains, each with different local security requirements. Much attention has been devoted to security issues and it is apparent that a high level of security is a fundamental prerequisite for Internet-based transactions, especially in the electronic commerce area.

As a consequence, the need for standard architectures and frameworks for developing such applications has arisen. The OMG[1] has specified the CORBA in response to these needs[2, 4, 6]. CORBA[8, 9] is a standard middleware supporting heterogeneous networks, designed as a platform-neutral infrastructure for inter-object communication. However, CORBA based security service specification[1, 10] itself does not provide any security mechanism[6]. The predefined attributes of CORBA security service have only limited validity and many security mechanisms do not provide sufficient security attributes.

In this environment, the number of new users and applications requiring authentication will continue to increase at a rapid rate. Clients must authenticate themselves to the Provider system. Clients must not be allowed to access arbitrary resources. Thus a more elaborated security infrastructure must be provided in different administrative domains.

This paper, based on CORBA security service specification[1], proposes the authentication service model supporting multiple domains with an extension to the

V.N. Alexandrov et al. (Eds.): ICCS 2001, LNCS 2073, pp. 413–422, 2001.
©Springer-Verlag Berlin Heidelberg 2001

Kerberos[13] authentication framework using public key cryptosystem(PKC)[15]. This proposed model, by PKC based certificates, assures the identification of a partner in the authentication of peer entities and the secure access to multiple domains in the authorization of underlying resources. Since our deployed Kerberos is extended to the authentication service model, it provides the flexibility of key management and the ability to leverage the public key certification infrastructure[7].

The organization of this paper is as follows. Section 2 presents the CORBA security service and the description of authentication service model in this paper. Section 3 describes the structure of the authentication service model supporting multiple domains in detail. Finally, Sections 4 and 5 contain a performance and conclusion, respectively.

2 Security Service Approaches

2.1 CORBA Security Service

The CORBA security service specification[1] is large in part due to the inherent complexity of security, and due to the fact that the security service specification includes security models and interfaces for application development, security administration, and the implementation of the security services themselves. All these and their interfaces are specified in an implementation of independent manner[9, 10]. So the interface of security service is independent of the use of symmetric or asymmetric keys, and the interface of a principal's credential is independent of the use of a particular certificate protocol.

The objective of this specification[1, 2] is to provide security in the ORB environment in the form of an object service. The focus lies hereby on confidentiality, integrity, and accountability. The model used by the CORBA security service specification involves principals that are authenticated using a principal authenticator object. Once authenticated, a principal is associated with a credential object, which contains information about its authenticated identity and the access rights under this identity. These credentials are then used in secure transactions, to verify the access privileges of the parties involved, and to register identities for actions that are audited or carried out in a non-repudiation mode.

The Client requests a remote object through a local reference. The Client's credentials are attached to the request by the security services present in the ORB, and sent along with the request over the transport mechanism in use. The remote object receives the request through its ORB, along with the Client's credentials. The target object can decide whether to honor the request or not, based on the access rights of the Client's identity.

When a request is received from a remote object, its right to access the resources requested can be checked through an access decision object, which can be used to compare the remote principal's credentials against access control settings. Typically there is not a default access control policy that the security services will enforce requests, since checking access rights are usually very application-specific.

2.2 Description of Authentication Service Model

On Internet, Certificate Authorities(CA) acts as trusted intermediaries when authenticating clients and servers. *Authentication Service Provider* in our proposed model is another form of trusted intermediary[14]. In a public key scheme, a CA issues a long-lived credential – a public key certificate. When both clients and servers have such certificates they can authenticate to each other without further reference to a CA. However, precisely because theses certificates are long-lived, some method is required to inform servers of revoked certificates. This can be done by requiring servers to check a certificate's current validity with the CA on each use of a certificate, or by distributing Certificate Revocation Lists(CRLs) to all servers periodically. In the proposed scheme, the Authentication Service Provider issues clients a short-lived credential, which must then be presented to obtain an access right for a particular server.

The Authentication Service Provider described in this paper is structured in layers; *Exchange Layer*, *Supporting Services Layer*. The Exchange Layer and Supporting Services Layer are responsible for the execution of CORBA security service by adding of a security and a message interceptor[3]. The Exchange Layer provides services for handling and packaging business items as well as transfer and fairness of mutual exchanges. The security attributes stored in each type of the basic objects determine the label of privilege that is required for the exchange.

In the Exchange Layer, we concentrate on the *Credential Service Block* that receives parsed messages by the exchange manager. This block handles a credential of each participant, thereby performing all secure invocations between a Client and a Provider. This block contains *Authentication Service, Authorization Service,* and *Session IDentifier Service*.

The Supporting Services Layer provides persistent object storage, a communication, and a cryptographic service. The communication services block supports communication between multiple domains entities. The architecture can support any other networks as well as the Internet. The cryptographic service block provides cryptographic primitives like message encryption or decryption, and key distribution. The object storage service block supports persistent and secure local storage of data; principal credential, required right, and domain access policy, etc.

3 Authentication Service Model Supporting Multiple Domains

Current distributed computing is really a federation of resources from multiple administrative domains, between the Kerberized domains[14] and the Non-Kerberized domains, each with its own separately evaluated and enforced security policies.

In this paper, we propose the Kerberos based Authentication Service Model supporting multiple domains. The assumption of our scheme is that only objects of users that have been authenticated must be authorized to use the underlying resources over multiple domains. And assume that system administrators will allow their systems to participate in our Authentication Service Model.

3.1 Specification of Kerberized Authentication Service with PKC

Every object of our model has a credential that contains security information including a key in Kerberized system, or contains granted rights in Non-Kerberized system. This will support the authentication for peer entities to perform the mutual negotiations, and the authorization to control the domain accesses.

For our Kerberized service, the authenticated key exchange protocol relies on the ElGamal key agreement[16, 17]. It is well known that the ElGamal cryptosystem for any family of groups for which the discrete logarithm is considered intractable. Part of the security of the scheme actually relies on the Diffie-Hellman assumption, which implies the hardness of computing discrete logarithms[17]. **Fig. 1** shows a protocol of ElGamal key agreement.

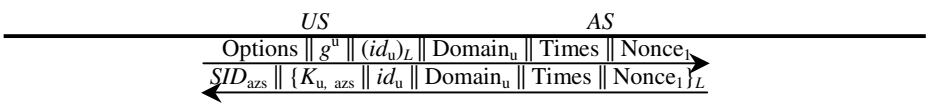
1. *B* picks an appropriated prime *p* and generator of Z^*_p and selects a random integer *b*, 1 *b* *p*-2. After computes ^{b} *mod p*, *B* publishes its public key (*p*, , ^{b}), keeping private key *b* secret.
2. *A* obtains an authentic copy of *B*'s public key (*p*, , ^{b}) and chooses a random integer *x*, 1 *x* *p*-2. Then *A* sends *B* follow protocol message and computes the key as *K* = (^{b})^{x} *mod p*.

   $A \rightarrow B$: ^{x} *mod p*
3. *B* computes the same key on receipt of the message as *K* = (^{x})^{b} *mod p*.

Fig. 1. ElGamal key agreement protocol

We define two important security methods we want to provide: the *authentication()* and the *authorization()*. The authentication() should normally be done by the User Sponsor(US). The Session IDentifier based the authorization() controls the access to the Providers, and the operation the Client wants to invoke. Finally, we declare the interface of authentication service provider that inherits the interfaces of the authentication() and the authorization().

With authentication() method, the US initiates the authentication exchange by requesting the Session IDentifier(SID_{azs}) from the Authentication Service Block(AS). This is necessary since the construction of the subsequent Authorization Service Phase requires the certificate of Provider(*CertP*). **Fig. 2** shows the process of obtaining the Session IDentifier (SID_{azs}) based on Kerberos[13] using PKC[15], between User Sponsor and Authentication Service Block.



$SID_{azs} = \{Flags \parallel K_{u, azs} \parallel Domain_u \parallel id_u \parallel Times\}_{Kazs}$

Fig. 2. Authentication Phase based on Kerberos using PKC

At the start of the protocol, there is one of the assumptions that AS has long-term secret and public key agreement keys *v* and g^v. Another assumption is that US possesses the public key necessary to verify certificates issued by Authentication Service Provider. In first request message, as shown in **Fig. 1**, US generates a random

number u and computes temporary public key agreement key g^u. The US then generates an encryption session key $L = (g^v)^u$ where g^v is the public key agreement key of the AS.

On receipt of the first message, AS does not know with whom he is communicating. AS computes $L = (g^u)^v$ and generates a session key $K_{u,\,azs}$ between US and Authorization Service Block(AZS). He then sends to US message encrypted using L together with Session IDentifier(SID_{azs}) to access AZS encrypted using the secret key of the AZS, K_{azs}.

Once the US has obtained SID_{azs}, it implies being authenticated by an authentication object, and can proceed to generate SID_p for service request. The message contains similar information to that in a traditional ticket request of Kerberos.

The method authorization() handles a Session IDentifier(SID_{azs}), the Provider(id_p) that the Client wants to access, and the name of the operation to invoke. We get Provider's name(id_p) from the object storage of Kerberized hosts and the principal(id_u) from the Session IDentifier(SID_{azs}). If the name of the Provider(id_p) the Client wants to invoke is among these, this is allowed to proceed. If the authorization succeeds, the operation and the returns are subsequently invoked on the Provider. If not, the CORBA system exception[8] 'NoPermission' is flagged. **Fig. 3** shows the process of obtaining another Session IDentifier (SID_p) based on Kerberos using PKC, between User Sponsor and Authorization Service Block.

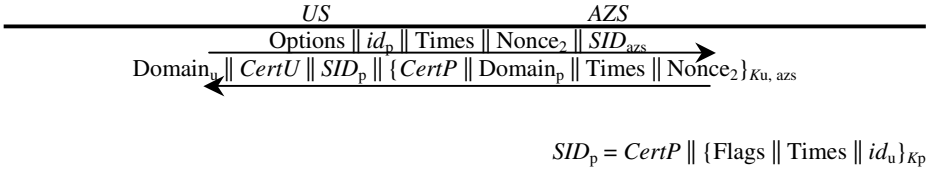


Fig. 3. Authorization Phase based on Kerberos using PKC

At the start of this phase, as shown in **Fig. 2**, there is an assumption that AZS has kept key escrow of Provider and has shared the domain($Domain_p$) with Provider. In first request message, US send to AZS Provider's identity(id_p)with SID_{azs} encrypted using secret key of AZS(K_{azs}).

On receipt of the first message, AZS decrypts the message using his secret key. It then retrieves the session key between US and AZS, which is found in the SID_{azs}. And then he generate the appropriate certificates required in the protocol($CertU$, $CertP$) and the Session IDentifier(SID_p) to access Provider. He send to US theses messages together with Session IDentifier(SID_p) encrypted using the secret key of the Provider(K_p).

The Session IDentifier(SID_p) received by the AZS is simply a conventional service ticket. At the start of this phase, as shown in **Fig. 3**, US has Provider's public key g^p in the certificate $CertP$ and Client's private key w in the certificate $CertU$. And then he computes an encryption session key $K_{u,\,p} = (g^p)^w$. In first request message, US send to PS the certificate of client($CertU$) and SID_p and the Authenticator which additional data needed as input to the payment scheme with encrypted using encryption session key $K_{u,\,p}$. **Fig. 4** shows the process of obtaining service, which the Client wants to

access based on Kerberos using PKC in the Open Authentication Service, between User Sponsor and Provider Sponsor(PS).

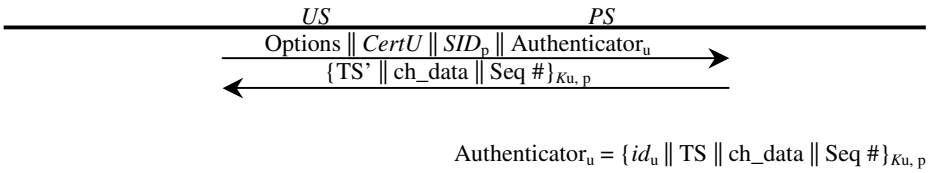


Fig. 4. Service Access based on Kerberos using PKC

On receipt of the first message, PS computes an encryption session key $K_{u,\,p} = (g^{w})^{p}$ then decrypts the encrypted Authenticator message. All operations from this point on can protocol per normal Kerberos operations.

3.2 Mechanism of Authentication Service Model for Kerberized Domains

On Kerberized domain, our client's request of Authentication Service Model needs initial objects to reference the security service, like it does for other CORBA based services. These objects are 'SecurityLevel2::UserSponsor(or ProviderSponsor)'[8, 10] and 'SecurityLevel2::Current'. **Fig. 5** shows a mechanism of Kerberized multiple domains access based on CORBA.

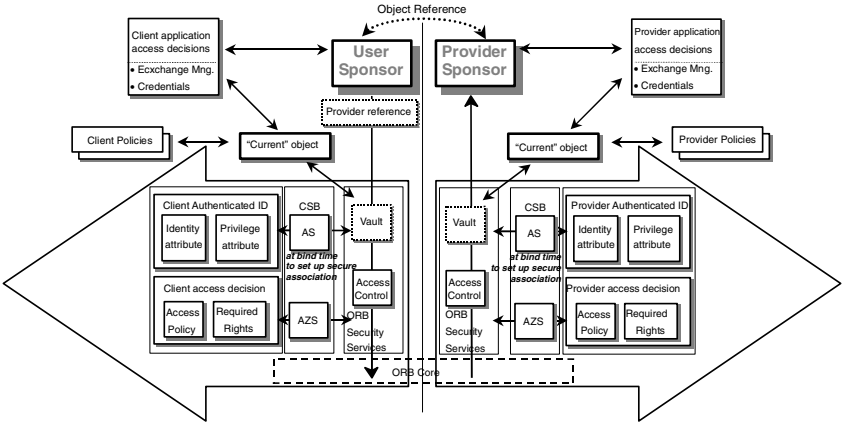


Fig. 5. Kerberized domains access based on CORBA

The UserSponsor calls on the PrincipalAuthenticator object, which authenticates the principal and obtains the Credential containing authenticated identity and privileges.

The Credential has to be created from its own certificate. This object holds the security attributes of a principal, e.g. authenticated identity and privileges. It is used by the application to create its own security information that later should be sent to the remote peer during the establishment of the secure association between the Client

and the Provider. The Current object and Vault object provide the system with interfaces to processing and storage resources. The use of these objects interfaces is encapsulated by UserSponsor(or ProviderSponsor) objects.

UserSponsor(or ProviderSponsor) provide a central mechanism for specifying policy for a set of like objects. For setting policy for instances, UserSponsor serves as location authorities for instances, supporting the binding of secure association.

Once an invocation passes an interaction point, the Secure Invocation Interceptor establishes a security context, which the Client initiating the binding can use 'Current::get_policy', to securely invoke the Provider designated by the object reference used in establishing the binding as 'get_policy'. After binding time, the Secure Invocation Interceptor is used for message protection, providing integrity and confidentiality of requests and responses, according to the quality of protection requirements specified in the active security context object.

Finally, as shown in figures of previous section, it can establish a fairness of exchanges through the Authentication Service Provider using items bundled as a CORBA object. Both the UserSponsor and the ProviderSponsor transfer Session IDentifiers(SID_{p}) before mutual exchanges occur. This prevents the Current from being transferred Current to illegal peers, and prevents the Clients from being given access to illegal peers.

3.3 Case of Authentication Service Model in Multiple Domains

For multiple domains, all communication of our model is done via Kerberos mechanisms. Thus cross-realm authentication[13] is immediately and transparently supported: UserSponsor only has to be performed once for each group of Kerberos realms that support cross-realm authentication with each other. The initial objects will automatically obtain SID_{azs}s for the other realms based on the existence of a valid SID_{azs} for a given host.

Our model is assumed with Kerberized domains basically, however, we must consider multiple domains, specifically integration of Kerberized(K_{sys}) and the Non-Kerberized domains(NK_{sys}). We propose the approach to issue temporary credentials to NK_{sys}'s request objects[11, 12]. Most of all participated NK_{sys} involve a single CA. In general, users in one domain are unable to verify the authenticity of certificates originating in a separate domain. However, our K_{sys}'s Authentication Service Provider issues a cross-certificate[4] based temporary certificate. A trust relationship between CAs of multiple domains and our Authentication Service Provider must be specified in order for users under CAs to interoperate cryptographically.

The essential component of our approach is a KProxy object for each NK_{sys} client's request. This KProxy securely holds the delegated credentials as K_{sys} client's in the local Kerberized system. Whenever the client's request from the NK_{sys} wants to create an object on client's behalf on its associated physical machine, the Authentication Service Provider creates the Current object that contains minimal permission. Provider's AS will only issue client's delegated credentials of that domain if client's valid temporary certificates are presented in the request. A delegated credential specifies exactly who is granted the listed rights, whereas simple possession of a bearer credential grants the rights listed within it. Then the Current performs a call

back to the KProxy for client to obtain a SID_{azs} for that particular Provider. **Fig. 6** shows the authentication mechanism of multiple domains access.

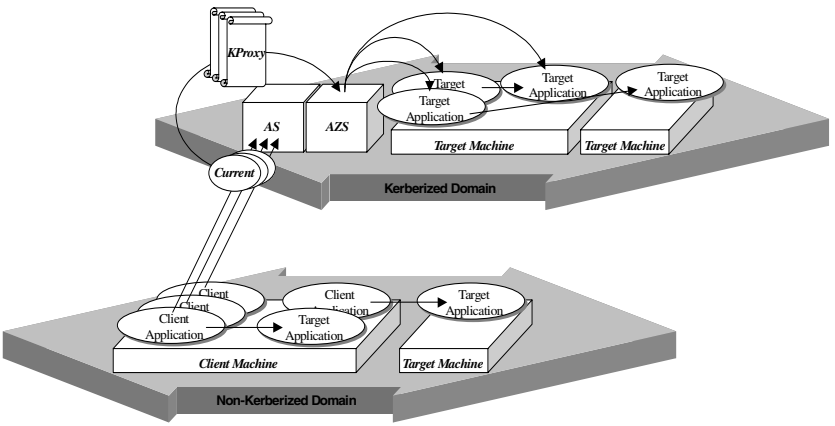


Fig. 6. Authentication mechanism of multiple domains access

After obtaining a SID_{azs}, Provider's AZS is obtained the attributes from the KProxy object by calling 'get_attributes'. Then all operations from this point on can mechanism per Kerberized operations for multiple domains.

The access control mechanisms of client's KProxy can be configured to issue the thread specific Credential obtaining the information to the thread of request execution. There might be more than one thread, every thread associated with a different set of security attributes. These can be accessed from the appropriate Current has to be used.

4 Security and Analysis

4.1 Security of Our Authentication Service Model

In distributed computing, the DCE[18] and the SESAME[5, 19] are the well-known security systems, based on Kerberos, of the Client/ Server architecture. As seen in the **Table 1**, our proposed Open Authentication Service shows better security features like the authentication for peer entities to perform the mutual negotiations and the fairness of exchange. Therefore it has the good advantages of interoperability with other security services.

Table 1. Security Analysis of Proposed Schemes

	DCE	SESAME	Authentication Service Model
Access control level	Application	Application	Application/ System
Authentication	Unilateral	Unilateral	Unilateral/ Mutual
Authorization policy	ACL based	ACL based	Label based Rule
Fairness of exchange	No	No	Yes
Flag of privilege type	Positive	Positive/ Negative	Positive
Grant/ Revoke privileges	Controlled by Server	Controlled by Server	Label based Rule
Scalability	Average	Average	High
Security policy domain	Server's domain	Server's domain	System Imposed
Suitability	Stable User base	Stable User base	Mandatory Controls

And then **Table 2** shows comparison of key distribution Scheme with traditional Kerberos. Our proposed Authentication Service Model provides session key establishment mechanism based on PKC.

Table 2. Comparison of Key Distribution Scheme with traditional Kerberos

	Kerberos	Authentication Service Model
Session Key Establishment between User and TGS(AS)	Symmetric Key Transport	ElGamal
Session Key Establishment between User and SGS(AZS)	Symmetric Key Transport	Symmetric Key Transport
Session Key Establishment between User and Service Provider	Symmetric Key Transport	ElGamal
Role of TGS(AS)	Symmetric Key Transport for User and Service Provider	Public Key Certificate Generation

5 Conclusion and Future works

We have proposed the CORBA based authentication model, and within that model we have presented flexible mechanisms to accommodate multiple domains. The goal of our system is to select resources for use by applications and securely coordinate execution of application in multiple domains, eliminate the need for the end-users to explicitly log on to each machine.

The authentication model that proposed in this paper is very critical in distributed computing, since it supports the protection of the high-level resources and the preservation of the security policies of the underlying resources that form the foundation of various domains, between the Kerberized domains and the Non-Kerberized domains. Using public-key cryptosystem we acquired the flexibility of key management and reliable session key generation between the Client and the Provider.

Research should be made on the efficient object system to support a distributed security mechanism, and offer a more elaborated security infrastructure. In addition, for a key management, a heterogeneous key distribution of session keys should be considered.

References

1. OMG, CORBA services: Common Object Security Specification v1.7(Draft), ftp://ftp.omg.org/pub/ docs/security/99-12-02.pdf, 2000.
2. Object Management Group. CORBA/ IIOP 2.3.1 specification, http://sisyphus.omg.org/technology/documents/corba2formal.htm, 1999.
3. OMG Security Working Group, OMG White Paper on Security, OMG Document, No. 9, 1996.
4. Menezes, Van Oorschot, Vanstone, Handbook of Applied Cryptography, 2nd Ed., pp.570-577, 2000.
5. Joris Claessens, A Secure European System for Applications in a Multi-vendor Environment, https://www.cosic.esat.kuleuven.ac.be/sesame/, 2000.6. A. Alireza, U. Lang, M. Padelis, R. Schreiner, and M. Schumacher, "The Challenges of CORBA Security", Workshop of Sicherheit in Mediendaten, Springer, 2000.
7. DSTC, Public Key Infrastructure RFP, ftp://ftp.omg.org/pub/docs/ec/99-12-03.pdf, 2000.
8. Robert Orfali, Dan Harkey, Client/ Server Programming with JAVA and CORBA, John Wiley & Sons, 1997.
9. Andreas Vogel, Keith Duddy, Java Programming with CORBA, 2nd Ed., John Wiley & Sons, 1998.
10. Bob Blakley, CORBA Security: An Introduction to Safe Computing with Objects, Addison Wesley, 2000.
11. M. Humphrey, F. Knabe, A. Ferrari, A. Grimshaw, “Accountability and Control of Process Creation in the Legion Metasystem”, Symposiu m on Network and Distributed System Security, IEEE, 2000.
12. A. Ferrari, F. Knabe, M. Humphrey, S. Chapin, and A. Grimshaw, “A Flexible Security System for Metacomputing Environments”, High Performance Computing and Networking Europe, 1999.
13. John T. Kohl, B. Clifford Neuman, Theodore Y. Ts’o, “The Evolution of the Kerberos Authentication Service”, EurOpen Conference, 1991.
14. Massachusetts Institute of Technology Kerberos Team, Kerberos 5 Release 1.0.5. http://web.mit.edu/kerberos/www/.
15. M. A. Sirbu, John Chung-I Chuang, “Distributed Authentication in Kerberos Using Public Key Cryptography”, Symposium on Network and Distributed System Security, IEEE, 1997
16. W. Diffie, M. E. Hellman, “New directions in cryptography”, IEEE Transactions on Information Theory, Vol. 22, No. 6, 1976.
17. T. ElGamal, “A public-key cryptosystem and a signature scheme based on discrete logarithms”, IEEE transactions on Information Theory, Vol. IT31, No. 4, 1985.
18. G. White and U. Pooch, “Problems with DCE Security Services”, Computer Communication Review, Vol. 25, No. 5, 1995.
19. T. Parker, D. Pinkas, SESAME V4 Overview, SESAME Issue1, 1995.

Performance and Stability Analysis of a Message Oriented Reliable Multicast for Distributed Virtual Environments in Java

Gunther Stuer, Jan Broeckhove, and Frans Arickx

Antwerp University
Department of Mathematics and Computer Sciences
Groenenborgerlaan 171, 2020 Antwerp, Belgium
gstuer@ruca.ua.ac.be

Abstract. The aim of this paper is to present the performance and stability analysis of a reliable multicast system. It has been optimized for use in distributed virtual environments and is implemented in Java. The paper will describe the characteristics of our reliable multicast implementation, as observed in our test environment. We will also compare with non-reliable multicast protocols.

1 Introduction

This paper describes the performance analysis of a message oriented reliable multicast protocol for distributed virtual environments.

In the construction of a Distributed Virtual Reality Environment (DVE), a reliable and efficient multicast protocol on the Internet is necessary [1]. First, lets put this paper in a broader perspective by describing its relevance in the development of a highly dynamical distributed virtual environment.

One of the bottlenecks in virtual environments has always been the availability of sufficient network bandwidth to allow the participating objects to communicate with each other [2]. With the introduction of multicasting this problem was partly solved, but most traditional multicast protocols have two drawbacks [3].

The first is that these protocols are based on *best effort approaches*, i.e. message delivery is not guaranteed. In order to achieve this guarantee, *reliable multicast* protocols were introduced [4]. Although there are already many such protocols, none is optimized for distributed virtual environments [5]. Since almost all of the existing reliable multicast protocols aim to send a relatively large data chunks (e.g. a file) from one source site, they are not suitable for application in a DVE. A DVE typically has many source sites, the size of the data is relatively small and there are a large number of messages. Also, to our knowledge, an implementation of such a protocol in Java, which might a priori have some performance and timing drawbacks, has not yet been attempted.

The second problem is that multicast groups are statically allocated [6]. With virtual environments one usually considers spatial criteria to divide the world in partitions, where each partition transmits its data on one multicast group. However,

V.N. Alexandrov et al. (Eds.): ICCS 2001, LNCS 2073, pp. 423-432, 2001.
© Springer-Verlag Berlin Heidelberg 2001

with dynamic environments this isn't sufficient anymore. Participants have a tendency to flock together and this leads to situations where some groups are very heavily used, while others are completely idle. Allocating multicast groups in a dynamical way can solve this problem [7]. Techniques that can be used for this include probing [8] and fuzzy clustering [9]. With these methods one can determine at runtime which participants should be put together in the same multicast groups at any given moment in time.

In the classification of reliable multicast protocols [3,12] the approach that we use is most closely related to the Transport Protocol for Reliable Multicast. When one classifies protocols on the basis of data buffering mechanisms [13], ours is a receiver-initiated approach, i.e. no acknowledgements of receipt (ACKs) are used. Instead, the receiver transmits a negative acknowledgement (NACK) if retransmission is needed, because there was an error in the message, because a skip in sequence numbers indicated a missing message or because a timeout has elapsed.

With this approach, two problems can arise: (1) a NACK implosion at the sender due to the detection of a missing packet by many receivers, and (2) buffer size limitations at the sender side. Indeed, in principle, the sender needs to keep all messages available for retransmission because a NACK may arrive at any time. One never knows whether all interested parties have successfully received the message. This leads to the fact that buffers should in principle be infinite.

Waiting a pseudo-random time interval before sending a NACK solves the first problem. Also, when a client is waiting to send a NACK and in the mean time it receives a NACK-request from another client for the same missing packets, it can drop his own request. The second problem is solved heuristically by assuming that messages are of no further interest after a configurable amount of time as indicated above. As indicated before, this is appropriate in VR applications.

2 Design and Implementation

We have implemented our reliable multicast protocol taking into account a number of design features and goals:

1. The protocol will be used in distributed virtual reality systems. From previous work [10] we know that this has some interesting implications.
   - The typical message size used in virtual reality applications is rather small (< 1kB) because once the viewers know what an object looks like and where it is positioned, one only needs to transmit the changes with respect to that information.
   - Because a frame rate of 30 Hz is considered acceptable, there is no point in sending more than 30 update messages per second.
   - When dead reckoning algorithms – i.e. determination of the current position on the basis of previous positions – are applied, an update rate of once per second will often suffice.
   - When a message doesn't arrive during the first few seconds after is has been sent, it has completely lost its relevance to the virtual world.

Based on the average message size and the maximum number of message sent per second, we can make a realistic prediction about buffer sizes and timeout windows that are key parameters in implementing the reliability in the protocol. It can also improve performance because we do not need to resize our buffers while in action.
Because we know the average and the maximum throughput, we can apply the Usage Parameter Control (UPC) algorithm a.k.a. leaky bucket algorithm. This algorithm can be used to control bandwidth usage. An example is its use in ATM networks [11].
And most importantly, we can relax the reliability criteria. It is appropriate for our problem to have the sender buffer messages, for possible retransmission, only for a *certain amount of time* and then discard them. The amount of time may vary depending on the type of message. This way we can assign each message an *importance factor*. Important messages should be kept longer in the buffer.

2. The protocol has to be implemented in JAVA. The main motivation is that the virtual reality system that is being designed will be implemented in Java. We chose Java because it has features that we want to use, such as multithreading, loading classes across the network and the write-once-run-everywhere strategy. However, choosing Java as the implementation language adds an extra difficulty because it is not the optimum choice for time sensitive applications.
3. In view of the need for easy maintainability and portability, we have put significant effort into obtaining a good design. We were rigorous in defining interfaces with ease of use in mind, and made extensive use of Design Patterns [14, 15].
4. The primary design goal for our VR system is that it has to be distributed. The termination of a node, due to crash or transmission failure over an extended period, should have minimal impact on the whole. As a consequence, the reliable multicast architecture also needs to be completely distributed. This means that every participating node should be able to operate independently of all others to ensure the functioning of the protocol and in particular its reliability.

For more details on the implementation and the particular algorithms that were used, we refer to [16].

3 Test Environment

The experiments were performed using 45 Pentium-III computers. Each has a 450 MHz processor, 64 MB memory and a 3Com 100 Mbit NIC. The network is a 100 Mbit Ethernet. The PCs are split in 5 groups, each group interconnected by a 100Mb HUB. A 100Mb switch connects these 5 groups. All computers have Windows98 as operating system and run Sun's JDK 1.2.2

In a third paper we will compare the reliable multicast protocol between different operating systems, different computers and different Java Virtual Machines.

4 Benchmarks for Java-Based Multicasting

In this paragraph we describe the artefacts we encountered while searching for the boundaries of multicasting in Java using our test environment described above.

The first thing one should wonder about when researching multicast behaviour is trying to find ways to determine how the Java API handles multicasting. The first experiment we designed had as sole purpose to check whether the Java-*send()* method is blocking or not. We checked this by sending a datagram every 10 milliseconds. From figure 1 one can see that the gap between the actual amount of bytes sent and the theoretical maximal amount increases with increasing datagram size. This gap signifies the time needed to actually send the datagram. And as such we can conclude that *send()* is a blocking method. If this would have been an asynchronous message, throughput per second would be proportional to the datagramsize as 10 milliseconds is more than time enough to send one datagram.

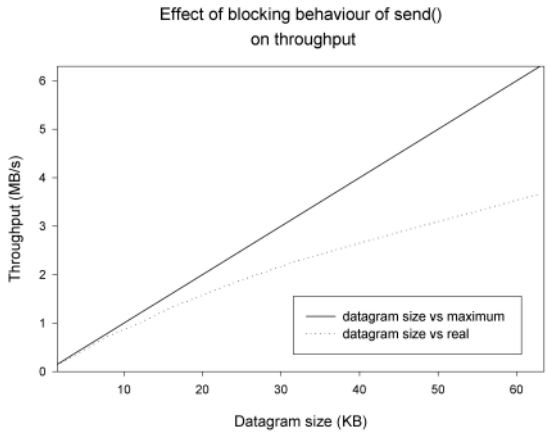


Fig. 1. The Java-send() operation is clearly a blocking operation. The larger the datagram sent, the larger the gap between the actual throughput and the maximal throughput.

In a typical VR application you have many nodes sending data to each other at the same time. In a second experiment we examined how, in our test environment, java multicast behaves when multiple servers are active at the same time. We have chosen to configure the servers as such that they each send a one-kilobyte packet every 33 milliseconds, the maximum values for VR applications. This way we can easily compare the results with those from our reliable multicast system. From figure 2 one can see that throughput is linear until we reach about 35 senders. After this, there are too many collisions and the increase in throughput flattens.

It is important to know what the ideal datagram size is. To discover this we designed the following experiment in which a server sends as many datagrams as possible of varying size. It has to be noted that multicast is build upon UDP and the maximum datagram size is 63 bytes. Figure 3 shows how throughput increases with increasing datagram size. This increase in throughput is dramatic for datagram sizes less than 8 kilobytes. After this, the increase is only average. From this graph one can

deduct three things. First, there isn't much sense in using datagrams larger than 8 kilobytes, secondly, when the datagrams are very small, throughput drops significantly.

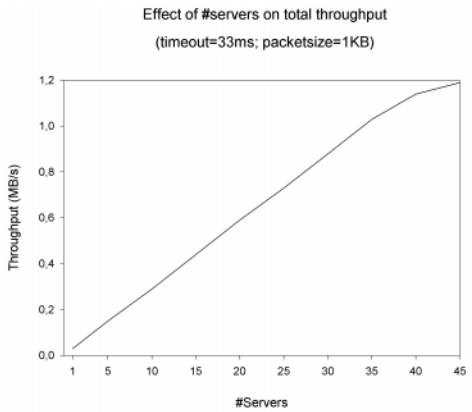


Fig. 2. As the number of servers increase, throughput increases linearly until there is a throughput of approximately 1.2 MB/s. This is due to the increasing amount of collisions and to the fact that it takes about 0.9 ms to handle one datagram.

Since in VR application, the typical message size is less than 1 kilobyte, one can see from this graph that in a configuration with one server, it is not possible to send more than approximately 5.6 MB/s. When multiple servers are active, the effect of collisions has to be taken into account (see figure 4).

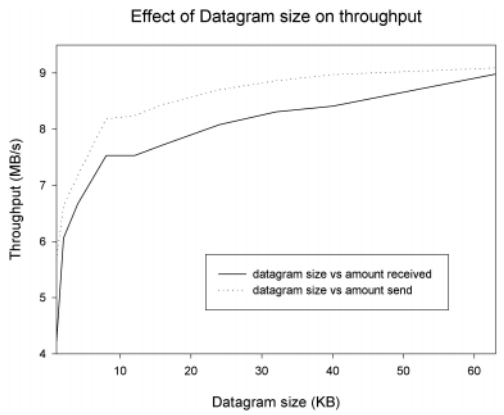


Fig. 3. For datagram sizes smaller than 8 KB, small increases result in major throughput gains. After this, throughput increases only moderately. Also note that there are about 10% missing datagrams. This is mainly due to the NIC being unable to handle all the datagrams fast enough.

The third observation one can make is that there is a discrepancy between the amount sent and the amount received. For normal multicast this isn't a real problem, but for reliable multicast systems, this can become a problem, as all datagrams have to be received.

The next experiment is used to determine the effect of pausing between consecutive sends. The server is configured to continuously send a 8KB packet and wait a configurable amount of time.

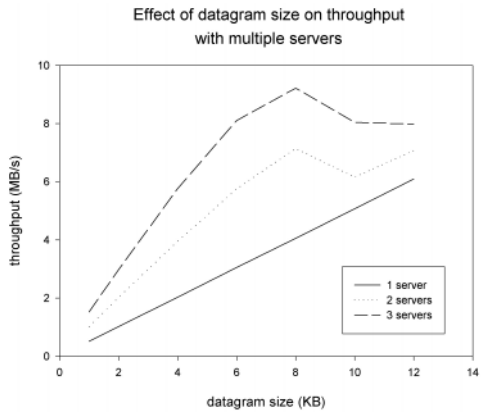


Fig. 4. When multiple servers are active, there is danger of collisions; we see a performance drop when datagrams become larger than 8 KB.

Figure 5 shows that without waiting, a maximum throughput of 7.5 MB/s can be achieved. This leads us to the conclusion that sending a 8KB packet takes about 0.9 ms. There the smallest amount of time we can wait is 1 ms, the slightest pause between sends, will halve the maximum throughput, as can be seen from figure 5. Since our multicast system uses the leaky bucket algorithm [11] to control congestion, we have to wait between consecutive sends. This implies that we won't be able to send more than about 520 messages a second. But for VR applications this is more than enough.

5 Performance Analysis

Since in a typical VR environment one has many participants, each sending its information, it's very important to construct an experiment that measures performance when many different servers are active. For this experiment we assumed the typical VR settings: the messages sent are one packet in size and each packet is 1KB. These messages are sent at a rate of 30 a second. Figure 6 shows how throughput evolves when we active more and more senders each sending 30 KB/sec. From figure 5 one can see that throughput increases until we reach 300 datagrams a second. This seems to be the maximum amount of datagrams our multicast system can handle. After this, throughput slowly degrades due to increasing datagram losses. When our VR environment needs more participating servers, we must either lower throughput, for example by using dead reckoning algorithms, or work with multiple multicast groups. An important note that has to be made is that performance and scalability is very dependant on the Operating System en Java Virtual Machine used. These observations will be discussed in an upcoming paper. Unfortunately, network conditions aren't always optimal. As such, it is important to determine the stability of

our protocol under problematic situations. For this we created an artificial error rate by dropping a certain percentage of all datagrams just before they are to be sent. For this experiment we had 1 server sending at a rate of 30 messages a second. Each message is 1 KB in size.

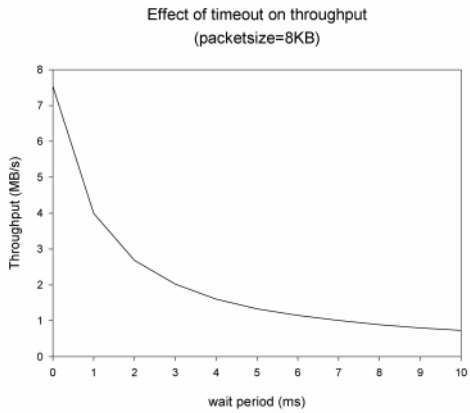


Fig. 5. When there is no pause between two consecutive sends, the application reaches its maximal throughput. This is approximately 7.6 MB. Slight pausing periods will severely lower total throughput.

From figure 7 we see that throughput remains very good, even under high error rates. This can be explained because we continuously put new messages in the system regardless whether the old ones were completely sent or not. A disadvantage of this is that the load on the servers increases with degrading networks.

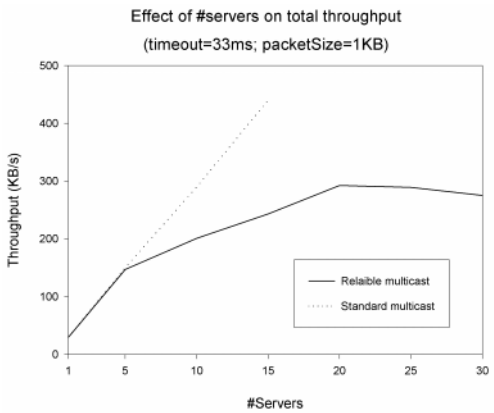


Fig. 6. Multiple servers, each sending at a rate of 30 datagrams a second, will initially increase the total throughput. At approximately 300 KB/s there is a breakpoint, which indicates that this is the maximal throughput the Reliable Multicast System can handle.

The third test was designed to see how different message sizes would influence throughput. From figure 3 we know that an 8 KB message would be ideal, but a

typical VR-message is only 1 KB and as such we would have a tremendous overhead. This is why we chose to fix the datagram size at 1 KB and test what the throughput will be when we send large messages consisting of multiple datagrams.

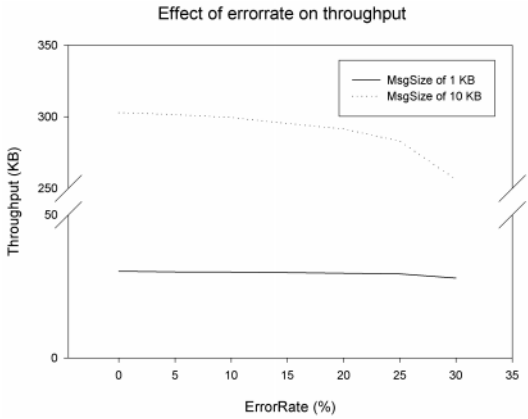


Fig. 7. For small messages, the error rate doesn't influence the total throughput because the server on sending new messages at the same rate. For large messages however, the server has to lower the injection of new messages because the maximal amount of sends per second is reached.

Once again we had a server sending at a ratio of 30 messages as second. This time however the message size varies from 1 till 80 packets, with each packet being 1 KB. As one can see from figure 8, there is almost no influence at all. From this we can conclude that handling large messages is as efficient as handling small ones. This test demonstrates that the used data structures work as expected.

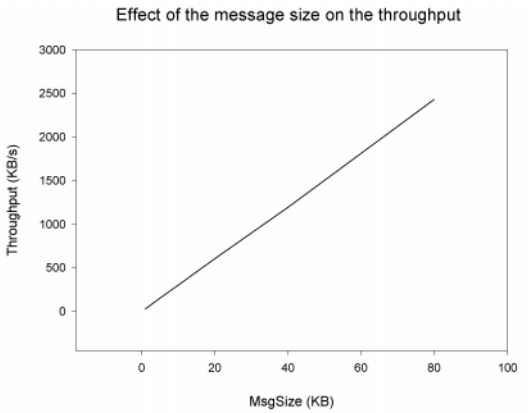


Fig. 8. Total throughput increases linearly with the message size. This indicates that the fragmentation and assembly algorithms work as expected.

The fourth and last experiment was designed to check how well our reliable multicast protocol would work when applied in other areas. For this we varied the amount of

messages sent per second. Each message is 1KB in size. As one can see from figure 9, the protocol remains efficient for large send frequencies. The flattening at the end of the curve indicates that our limit is at 400 sends a second. Which is very good when you consider that the maximum for standard, non-reliable, multicast is about 520 messages a second.

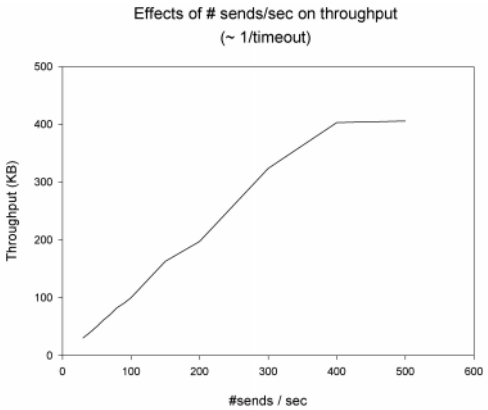


Fig. 9. As the number of sends a second increase, the throughput increases with it. From a send rate of approximately 400 messages a second there is stagnation in the throughput. This indicates that this is the maximum send rate of the Reliable Multicast System.

6 Future Work

This reliable multicast protocol is an important element in a much bigger project: the creation of a highly dynamical distributed virtual environment.

The next step will be minimizing the problems still remaining as stated above. The design and implementation of the probe classes [16], which will strongly decrease the total amount of messages sent when new objects enter the virtual world, will be considered in the near future. This technique is based upon the idea of sending chunks of code to the participants instead of data. This code will be able to negotiate whether the two objects are interested in each other, or not. If they are, the multicast groups on which they transmit their data will be exchanged.

As a next step, we will implement the fuzzy clustering algorithm [9] to dynamically allocate a fixed set of multicast groups to all participating objects. The most challenging task will be to define the criteria, which will determine which objects should be grouped together at a certain moment in time. To make this mechanism as flexible as possible, the criteria will be described in XML [17].

7 Conclusions

We think that we can safely conclude that the current version of the reliable multicast protocol for distributed virtual environment written in Java meets its performance targets inherent to the design goals. Due to the constantly improving performance and growing feature set of Java, the construction of a full blown distributed virtual reality system becomes more and more plausible. The protocols presented in this and future papers can only help to make the communications scheme more reliable, dynamic, performing and scalable.

References

1. Fumiaki Sato, Kunihiko Minamihata, Hisao Fukuoka, Tadanori Mizuno, "A Reliable Multicast Framework for Distributed Virtual Reality Environments", Proceedings of the 1999 International Workshop on Parallel Processing.
2. Michael J. Zyda, "Networking Large-Scale Virtual Environments", Naval Postgraduate School, Monterey, California, USA.
3. Katia Obraczka, "Multicast Transport Protocols: A survey and taxonomy", IEEE Communications Magazine, January 1998, pp. 94-102.
4. Kara Ann Hall, "The implementation and evaluation of reliable IP multicast", University of Tennessee, Knoxville, USA, 1994.
5. Kenneth P. Birman, "A Review of experiences with reliable multicast", Software – Practice and Experience 29(9), 741-774 (1999)
6. Chris Greenhalgh, "Dynamic, embodied multicast groups in MASSIVE-2", Technical Report NOTTCS-TR-96-8, University of Nottingham, UK, 1996.
7. Chris Greenhalgh, "Spatial Scope and Multicast in Large Virtual Environments", Technical Report NOTTCS-TR-96-7 University of Nottingham, UK, 1996
8. Gunther Stuer, Jan Broeckhove, Frans Arickx, "A message oriented reliable multicast protocol for a distributed virtual environment", ICSE'99 (CS-163)
9. C. Looney, "Fuzzy Clustering: A new algorithm", ICSE'99 (CS-115)
10. Kris Demuynck, Jan Broeckhove, Frans Arickx, "The VEplatform system: a system for distributed virtual reality", Future Generation Computer Systems 14 (1998), pp. 193-198.
11. William Stallings, "Data & Computer Communications, 6th edition", ISBN 0130843709, pp 405
12. B. Sabata, M.J. Brown, B.A. Denny, "Transport Protocol for Reliable Multicast: TRM", Proc. IASTED International conference Networks, January 1996, pp. 143-145.
13. Brian Neil Levine, J.J. Garcia-Luna-Aceves, "A comparison of reliable multicast protocols", Multimedia Systems 6 (1998), pp. 334-348
14. Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides, "Design Patterns", Addison-Wesley.
15. Gunther Stuer, Jan Broeckhove, Frans Arickx, "Design and Implementation of a Reliable Multicast Protocol for Distributed Virtual Environments written in Java", submitted to the EuroMedia 2001 Conference.
16. Gunther Stuer, Frans Arickx, Jan Broeckhove, "A message oriented reliable multicast protocol for J.I.V.E.", Parco99, Parallel Computing – Fundamentals & Applications, pp. 681-688.
17. S. Laurent, "Building XML Applications", Osborn McGraw-Hill.

A Secure and Efficient Key Escrow Protocol for Mobile Communications

Byung-Rae Lee, Kyung-Ah Chang, and Tai-Yun Kim

Department of Computer Science and Engineering, Korea University,
1, 5-ga, Anam-dong, Sungbuk-ku, Seoul, 136-701, Korea
{brlee, gypsy93, tykim}@netlab.korea.ac.kr

Abstract. In this paper we present secure and efficient key escrow protocols that guarantees escrow secrecy, public verifiability, and robustness for mobile telecommunications systems. We present a new construction for key escrow scheme, which compared to previous solutions by Chen, Gollman, Mitchell and later by Martin, achieves improvements in efficiency and security. We proposed a new key escrow protocol, designed for the case where the pair of communicating users are in different domains, in which the pair of users and all the third parties jointly generate a session key for end-to-end encryption.

1 Introduction

In modern secure telecommunications systems there are likely to be two contradictory requirements. On the one hand users want to communicate securely with other users, and on the other hand governments have requirements to intercept user traffic in order to combat crime and protect national security. A key escrow system is designed to meet the needs of both users and governments, where a cryptographic key for user communications is escrowed with an escrow authority (or a set of authorities) and later delivered government agencies when lawfully authorized.

When users communicate internationally, there is a potential requirement to provide the law enforcement agencies of all the relevant countries, e.g. the originating and destination countries for the communication, with warranted access to the user traffic. For example, mobile telecommunications systems might provide an end-to-end confidentiality service to two users in two different countries, and law enforcement agencies in both these countries might independently wish to intercept these communications.

In this paper we suppose that, in some environments where international key escrow is required, Trusted Third Parties (TTPs, acting as escrow authorties) may not be trusted individually to provide proper contributions to an escrowed key and to

V.N. Alexandrov et al. (Eds.): ICCS 2001, LNCS 2073, pp. 433-443, 2001.
© Springer-Verlag Berlin Heidelberg 2001

reveal the key legally, and users also may not be trusted to provide proper contributions to an escrowed key. We refer to domains instead of countries throughout.

Requirements for key escrow in an international (i.e. a multi-domain) context has been described by Chen, Gollman and Mitchell [1]:

1. No domain can individually control the generation of an escrowed key, and hence the escrowed key cannot be chosen by entities in only one domain and then transferred to the other domain.
2. The interception authorities in any domain can gain access to an escrowed key without communicating with any other domain, i.e. the key has to be capable of being escrowed in all relevant domains independently.
3. The entities in any domain can ensure the correctness and freshness of the escrowed key.

In this paper we present secure and efficient key escrow protocols for mobile telecommunications systems. Major achievements of our protocols are public verifiability on the correctness in recoverability of the session key and robustness. In our scheme the faulty behavior of any reasonably sized coalition of TTPs can be tolerated. The user just sends a particular ElGamal encryption [2,3] of the private key plus a proof that it indeed is a valid encryption of a discrete logarithm of the public key. In our scheme user needs to communicate only $O(k)$ bits and to perform the work $O(k)$. Moreover, the work of a TTP is also $O(k)$.

Our new key escrow protocols work in the model set forth by Chen et al. [1], where two entities A and B, located in mutually mistrusting domains, establish a shared session key based on Diffie-Hellman key exchange.

In our model, each user escrows his private key by posting an ElGamal encryption to the bulletin board. The encryption does not reveal any information on the private key itself but it is ensured by a proof of knowledge that the encryption indeed contains a valid private key. Due to the verifiable encryption technique and the bulletin board communication model, the recoverability of the user's private key can be verified by observer. This ensures public verifiability.

To prevent deviant users from obtaining a 'shadow-key', users and third parties jointly generate an escrowed key.

1.1 Properties for Our Key Escrow Protocols

Let us state and discuss the properties of the key escrow considered in this paper. Escrow secrecy and robustness come from the robust threshold cryptosystem [5,6,7]. Public verifiability comes from the ElGamal encryption and the proof of knowledge of Chaum and Pedersen [8]. Other two properties achieved from the joint generation of the session key.

Escrow Secrecy For any group of less than t TTPs it must be infeasible to recover the session key.

Public Verifiability Everybody, e.g. any network provider, can verify the correctness in recoverability of the session key.

Robustness The faulty behavior of any reasonably sized coalition of TTPs can be tolerated. In key escrow protocols this includes that no user can disrupt the key escrow scheme; in other words, any cheating user can be detected and discarded.

Resistance to Shadow Public Key Attack An escrowed key in the proposed protocols is a function of contributions from the user and all relevant TTPs. This property prevents two users abusing the mechanism by using the 'shadow-public-key' attack proposed by Kilian and Leighton [9].

Session Key Freshness If any entity updates its contribution using a fresh and random number, the key will be fresh and random. Any entity, which has updated his contribution to the session key can verify the freshness of the key.

1.2 Previous Work

The proposed protocols build directly upon international key escrow protocols of Chen, Gollman, Mitchell [1] and Martin [4], where a verifiable secret sharing (VSS) scheme used as an important primitive. In their protocol, each user A and B, located in mutually mistrusting domains, distributes shares of the private key to the TTPs using VSS respectively. A general problem with such a solution is that TTPs can verify the validity of only their own shares, but they cannot know whether other TTPs have also received valid shares. It may open the possibility for disputes: On the one hand a dishonest user may just skip sending a message to a TTP, while on the other hand a dishonest TTP may claim not to have received a message.

Their protocol lacks of explicit mechanisms to prevent malicious TTPs, e.g, TTPs submitting incorrect shares during the recovery procedure. This problem is usually dealt with implicitly though. In the schemes of [1,4] it suffices that the TTPs simply releases their shares. Subsequently the released shares may be verified by anybody against the output of the distribution protocol.

In their scheme major communication costs are generation of shares of a secret sharing scheme, and communication between users and their home TTPs and between TTPs from different domains. For security parameter k, each user has to perform $O(nk)$ modular multiplication and the TTPs also have to perform $O(nk)$ modular multiplication to transfer secrets to the other set of TTPs in a different domain using VSS.

Recently, Young and Yung [10] presented the scenario and solution for software key escrow. In their approach each user generates its own key pair, and registers with a certification authority. The user must also encrypt shares of the secret key for a specified group of trustees. The key pair is only accepted, if the user also provides a proof that the encrypted shares indeed correspond to the registered public key. Wenbo [11] described a partial key escrow scheme based on Stadler's verifiable encryption [12] and the robust threshold cryptosystem [7].

1.3 Our Contributions

The main contribution of this paper is a simple and fair international key escrow protocol based on robust threshold cryptosystem [5,6,7] and publicly verifiable encryption. To this end, we will employ fault-tolerant threshold cryptosystems instead of verifiable secret sharing scheme. In our scheme there will be only one public key for which the matching private key is shared among the TTPs using threshold cryptography techniques. Each user gives to the home TTPs a single escrowed encryption of the private key and decryption must be carried out in threshold collectively by a set of TTPs. Unlike previous schemes based on Chen's approach, however we will achieve robustness w.r.t. The correctness of the decryption will be assured, even in the presence of malicious TTPs. To achieve robustness against faulty TTPs we used the proof of knowledge of Chaum and Pedersen [8].

Recall that [1,4] requires the availability of private channels from the user to each of the home TTPs individually. However, communication over the private channels is clearly not publicly verifiable. We replaced the private channels by the ElGamal public key encryption. In our scheme, Each user sends an ElGamal encryption of the private key and a proof that TTPs can recover the private key. This is achieved via applying an ElGamal encryption with the proof of knowledge. The proof prevents the users from casting invalid encryption, and should be such that no information whatsoever leaks about the actual private key contained in an encryption.

Another contribution of this paper is a fair key escrow scheme in which the complexity of the user's protocol is linear in the security parameter k. This comprises the computational as well as the communication complexity (in bits). Each user needs to communicate only $O(k)$ bits and to perform $O(k)$ modular multiplication. Moreover, the dominating factor for the work of a TTP is k. Compared to the scheme of [1,4], we thus achieve a reduction of the work for each participant by a factor of n.

A session key for end-to-end encryption is established based on Diffie-Hellman key exchange. An escrowed key is a function of contributions from the user and all relevant TTPs to prevents two users abusing the mechanism by using the 'shadow-public-key' attack as shown in [1].

The main scheme presented in the paper is based on the security of the ElGamal encryption scheme (which is related to the difficulty of the Decision Diffie-Hellman problem).

1.4 Organization of the Paper

The remainder of the paper is subdivided as follows. In Section 2, verifiable encryption techniques are described. In Section 3, We explain the robust threshold cryptosystem. We propose a new key escrow protocol in Section 4. In our protocol two sets of TTPs, one group in each domain, are used as escrow authorities for two users. In Section 5, we conclude by giving the brief review on proposed protocols.

2 Verifiable Encryption

In this section, we will describe a verifiable encryption technique based on the ElGamal encryption and the proof of knowledge by Chaum and Pedersen.

2.1 Bulletin Board

The communication model required for our key escrow protocol is a public broadcast channel with memory, which is called a bulletin board. All communication through the bulletin board is public and can be read by any party (including passive observers). No party can erase any information form the bulletin board, but each active participant can append message to its own designated section.

2.2 Proofs of Knowledge for Equality of Discrete Logarithms

We will use the protocol by Chaum and Pedersen [8] as a subprotocol to prove that $\log_g x = \log_h y$, where by a prover shows possession of an Z_q satisfying $x = g$ and $y = h$.

1. The prover sends $a = g^w$ and $b = h^w$ to the verifier, with $w \ _R \ _q$.
2. The verifier sends a random challenge $c \ _R \ _q$ to the prover.
3. The prover responds with $r = w \quad c$
4. The verifier checks that $a = g^r x^c$ and $b = h^r y^c$.

It is well known that the above protocol is zero-knowledge only against the honest verifier. In order to make the protocol non-interactive, the verifier will be implemented using the Fiat-Shamir heuristic [13] which requires a hash function. In this case security is obtained for the random oracle model.

2.3 Double Exponentiation and Double Discrete Logarithms

Let p be a large prime so that $q = (p \quad 1)/2$ is also prime, and let $h \quad _p$ be an element of order q. Let further G be a group of order p, and let g be a generator of G so that computing discrete logarithms to the base g is difficult.

Our scheme will make use of double exponentiation. By double exponentiation with bases g and h we mean the function

$$_p \quad G : x \quad g^{(h^x)}$$

By the double discrete logarithm of g to the bases g and h we mean the unique $x \quad _q$ with

$$y = g^{(h^x)}$$

if such an x exists.

2.4 Verifiable Encryption of Discrete Logarithms

Our key escrow scheme is identical ElGamal's public key system [2], which is a variation of the Diffie-Hellman key-exchange protocol [3].

First, each TTP randomly chooses a secret key $z \quad _q$ and publishes his public-key $y = h^z \pmod p$. To encrypt a message $m \quad _p^*$ with the public-key y, the user randomly chooses $\quad _q$ and calculates the pair

$$\left(h \quad , y \quad m \quad ^1\right) \pmod p .$$

The ciphertext (A, B) can be decrypted by the recipient by calculating

$$m = A^z / B \pmod p .$$

Let us now describe a protocol for verifying that a pair (A, B) encrypts the discrete logarithm of a public element $P = g^s$ of the group G. It is based on the fact that if (A, B) is equal to $\left(h \quad , y \quad s \quad ^1\right) \pmod p$ for any $\quad _q$ then

$$P^B = g^{sB} = g^{(y \quad)} .$$

By the discrete logarithm of $g^{\{y\}}$ to the base g, we can now immediately obtain an efficient proof of knowledge for the following relation:

$$\log_h A = \log_y \left(\log_g P^B\right).$$

To prove that a pair (A, B) encrypts the discrete logarithm of a public key P, we have the efficient proof of knowledge by Chaum and Pedersen, described above.

3 Robust Threshold ElGamal Cryptosystem

The robust threshold cryptosystems described here is a slight variation from [5,6,7]. The main protocols of a threshold system consists of a key generation protocol to generate the private key jointly by the receivers, and decryption protocol to jointly decrypt a ciphertext without explicitly reconstructing the private key.

Key Generation The TTPs will execute a key generation protocol due to Pedersen [5] or rather improvement by [14]. The result of the key generation protocol is that each TTP T_i will possess a share $z_i \quad Z_q$ of a secret z. The TTPs are committed to these shares as the values $y_i = h^{z_i}$ are made public. Furthermore, the shares z_i are such that the secret z can be reconstructed from any set of z shares using appropriate Lagrange coefficients, say:

$$z = \sum_i z_i \, \lambda_{i,\ldots}, \quad \lambda_{i,\ldots} = \prod_{l \in \{i\}} \frac{l}{l-i}.$$

The public key $y = h^z$ is announced to all participants in the system.

Decryption To decrypt a ciphertext $(A, B) = \left(h \ldots, y \ldots m^{-1}\right)$ without reconstructing the secret z, each TTP broadcasts $w_i = A^{z_i}$ and proves in zero-knowledge that

$$\log_h y_i = \log_A w_i$$

as described in Section 2.2. Let denote any subset of t TTPs who passed the zero-knowledge proof. Now the plaintext can be recovered as

$$m = \prod_i w_i^{i,} \Big/ B .$$

Note that no single participant learns the secret z, and that the value of z is only computationally protected. The above protocol assures that the decryption is correct and successful even if up to $n \quad t$ TTPs are malicious or fail to execute the protocol.

4 The Key Escrow Protocol

Given the primitives of the previous section we now assemble a simple and efficient key escrow protocol, where two sets of TTPs, one group in each domain, are used as multiple authentication servers for the users and key escrow agencies for the interception agencies.

In this protocol, A and B are users in different domains. There are n TTPs $T_1,\cdots,T_n$ working for A as escrow authorities (in A's domain), and l TTPs $U_1,\cdots,U_l$ working for B as escrow authorities (in B's domain). Users and interception agencies do not communicate with TTPs outside their domain. Each set of home TTPs has agreed a common secret key (K_T and K_U respectively) and a key generation function f. This function takes as input the identity of two users, a clock C and a secret TTP key, and outputs a random secret. Let $f(A,B,C,K_T) = s_T$ and $f(A,B,C,K_U) = s_U$. We assume that clocks are synchronized among the set of TTPs. Before the protocol starts users do not share any secret.

In the following protocol, two sets of TTPs $T_1,\cdots,T_n$ and $U_1,\cdots,U_l$ assist two users A and B respectively to establish a session key K_{AB}. Each set of third parties escrow the key collectively.

Initialization As part of the initialization the designated parties generate the system parameters p,q,g,G as described in Section 2.3. Two sets of TTPs $T_1,\cdots,T_n$ and $U_1,\cdots,U_l$ will execute a key generation protocol respectively as described in Section 3.

Key Establishment The protocol consists of the following steps:

1. A chooses a random private key s_A, generates an encryption (x_A, y_A) of s_A using the public key of T_i, $1 \quad i \quad n$ accompanied by a proof of knowledge. She also publishes the public key $P_A \ (= g^{s_A})$.
2. B chooses a random private key s_B, generates an encryption (x_B, y_B) of s_B using the public key of U_j, $1 \quad j \quad l$ accompanied by a proof of knowledge. She also publishes the public key $P_B \ (= g^{s_B})$.
3. T_i, $1 \quad i \quad n$ verifies the encryption (x_A, y_A). T_i, $1 \quad i \quad n$ generates a secret s_T, calculates $P_{AT} \ (= P_A^{s_T})$, and sends P_{AT} to U_j, $1 \quad j \quad l$.

4. U_j, $1 \quad j \quad l$ verifies the encryption (x_B, y_B). U_j, $1 \quad j \quad l$ generates a secret s_U, calculates $P_{BU} \ (= P_B{}^{s_U})$, and sends P_{BU} to T_i, $1 \quad i \quad n$.
5. T_i, $1 \quad i \quad n$ calculates $P_{BUT} \ (= P_{BU}{}^{s_T})$ and sends P_{BUT} to A.
6. U_j, $1 \quad j \quad l$ calculates $P_{ATU} \ (= P_{AT}{}^{s_U})$ and sends P_{ATU} to B.
7. Finally, A and B separately compute a session key as:

$$K_{AB} = (P_{BUT})^{s_A} = (P_{ATU})^{s_B} = g^{s_A s_T s_U s_B}.$$

Key Recovery Any set of t TTPs in each domain can recover the session key separately:

1. T_i, $1 \quad i \quad n$ jointly execute the decryption protocol as described in Section 3 for (x_A, y_A) to obtain s_A. T_i, $1 \quad i \quad n$ can recover the session key K_{AB} with the knowledge of P_{BUT} and s_A.
2. U_j, $1 \quad j \quad l$ jointly execute the decryption protocol as described in Section 3 for (x_B, y_B) to obtain s_B. U_j, $1 \quad j \quad l$ can recover the session key K_{AB} with the knowledge of P_{ATU} and s_B

In this protocol any set of t TTPs in each domain can compute the session key K_{AB} established between A and B, but no group of $t \quad 1$ or less TTPs can recover the session key. The fact that all entities are involved in the key generation process helps make it more difficult for deviant users to subvert the escrowed key by using a hidden 'shadow-key'. In addition, no third party can force A or B to accept a wrong message unless all the third parties are colluding.

The performance of the protocol is as follows. The work for an user is clearly linear in k, independent of the number of TTPs. The work for the TTP is $O(nk+k)$ for T_i, $1 \quad i \quad n$ and $O(lk+k)$ for U_j, $1 \quad j \quad l$ respectively. Since key generation protocol is an one time operation of system setup procedure, the work for the TTPs is actually $O(k)$.

Our result can be summarized in the following theorem.

Theorem 1 If the ElGamal cryptosystem is semantically secure, then our key escrow protocol provides escrow secrecy, public verifiability, robustness and resistance to shadow-public-key attack.

Proof Escrow secrecy is guaranteed by the security of the ElGamal cryptosystem used to encrypt the private key. This is true because we assume that no more than $t \quad 1$ TTPs conspire, since t TTPs can reconstruct the private key used in the scheme. Any group of at least t TTPs can compute the user's private key (by the properties of the threshold ElGamal cryptosystem discussed in Section 3 above).

Public verifiability is achieved because any observer can check the proof of knowledge for the encryption, since those are made non-interactive.

Robustness with respect malicious users is achieved by means of the verifiable encryption technique, which ensures that users cannot submit bogus encryption. Ro-

bustness with respect to at most $n - t$ malicious authorities is inherited from the robustness of the key generation and decryption protocols.

Finally, shadow-public-key attack is prevented by joint generation of the session key. □

For the non-interactive version of the scheme based on the Fiat-Shamir heuristic, the result holds in the random oracle model.

5 Concluding Remarks

We have shown secure and efficient key escrow protocols for mobile communications based on the robust threshold cryptosystem and publicly verifiable encryption. In our scheme the work for the user is minimal and independent of the number of TTPs. The work for the TTPs is reduced accordingly. Another important difference is that due to the use of a threshold cryptosystem we achieve robustness in a strong sense. Proposed escrowed key agreement protocol, which meets possible requirements for international key escrow, where different domains do not trust each other.

We must note that it is difficult for any key escrow system to force two users to use only the current escrow key for their end-to-end encryption if the users share a secret or can use their own security system.

References

1. L. Chen, D. Gollmann, and C.J. Mitchell, "Key escrow in mutually mistrusting domain," Cambridge Workshop on Security Protocols, LNCS, vol.1189, pp.139-153, Springer-Verlag, 1997.
2. T. ElGamal, "A public key cryptosystem and a signature scheme based on discrete logarithms," IEEE Transactions. on Information Theory, vol.IT-31, no.4, pp.469-472, 1985.
3. W. Diffie, and M. Hellman, "New Directions in Cryptography," IEEE Transactions on Information Theory, November, 1976.
4. K. M. Martin, "Increasing efficiency of international key escrow in mutually mistrusting domains," 6th IMA Conference on Cryptography and Coding, LNCS, vol.1355, pp.221-232, Springer-Verlag, 1997.
5. T. Pedersen, "A threshold cryptosystem without a trusted party," In Advances in Cryptology-Eurocrypt '91, vol.547 LNCS, pp.522-526, Springer-Verlag, 1991.
6. T. P. Pedersen. Distributed Provers and Verifiable Secret Sharing Based on the Discrete Logarithm Problem, PhD thesis, Aarhus University, Computer Science Department, Aarhus, Denmark, March 1992.
7. R. Cramer, R. Gennaro and B. Schoenmakers. "A secure and optimally efficient multi-authority election scheme," In Advances Cryptology – Eurocrypto'97, LNCS vol.1233, pp.103-118, Springer-Verlag, 1997.
8. D. Chaum, and T. P. Pedersen, "Wallet databases with observers" In Advances in Cryptology-Crypto'92, vol.740 LNCS, pp.89-105, Springer-Verlag, 1993.
9. J. Kilian, and F. T. Leighton, "Fair Cryptosystem Revisited," In Advances Cryptology – Crypto'95, LNCS 963, pp.208-221, Springer-Verlag, 1995.

10. A. Young, and M. Yung, "Auto-Recoverable Auto-Certifiable Cryptosysem," In Advances Cryptology – Eurcrypt'98. LNCS, Springer-Verlag. 1998.
11. W. Mao, "Publicly Verifiable Partial Key Escrow," Proc. ACISP, pp.240-248, Springer-Verlag, 1997.
12. M. Stadler. "Publicly Verifiable Secret Sharing," In Advances Cryptology – Eurocrypt'96 LNCS vol.1070, pp.190-199, Springer-Verlag, 1996.
13. A. Fiat, and A. Shamir, "How to prove yourself: Practical solutions to identification and signature problems," In Advances Cryptology-Crypto '86, vol.263 LNCS, pp.186-194, Springer-Verlag, 1987.

High-Performance Algorithms for Quantum Systems Evolution

Alexander V. Bogdanov, Ashot S. Gevorkyan, and Elena N. Stankova

Institute for High Performance Computing and Data Bases,
Fontanka, 118, 198005, St-Petersburg, Russia,
bogdanov@hm.csa.ru, ashot@fn.csa.ru, lena@fn.csa.ru

Abstract. We discuss some new approach for derivation of computational algorithms for evolution of quantum systems. The main idea of the algorithm is to make in functional or path representation of quantum observable the transformation of variables, that makes the phase of path integral to be quadric functional. Thus the new representation for observable is reduced to standard multidimensional integral and the solution, for every point in coordinate space of that integral, of the first order partial differential equation system. That problem, although still difficult, can be very effectively parallelyzed. The approach is illustrated with the help of important example - the scattering in molecular system with elementary chemical reactions. The use of proposed approach show substantial speed-up over the standard algorithms and seems to be more effective with the increase of the size of the problem.

1 Introduction

Even with increase of the power of computers, used for quantum collision problem analysis nowadays, we do not feel the drastic increase of computational possibilities, especially when the number of nodes in computations is high. After some analysis it does not seem so strange since with the increase of number of parallel computational processes the price, which we pay for exchange of data between processes, is becoming more and more heavy load, that make the increase of number of nodes ineffective. That is why it is very important in deriving the algorithms to minimize the data exchange between computational processes. The standard algorithms for quantum scattering calculations are very ineffective for complex systems since in many realistic situations strong coupling between large amount of interacting states should be taken into account additionally to nontrivial problems with asymptotic boundary conditions. Among many candidates for alternative approach one of the most promising is the reduction of the original formulation to Feynman's path integral. One of the firsts such effective algorithms was proposed in 1991 ([1]). But due to some mathematical properties of path integrals this approach could be effectively used only for finite time interval evolution calculation. Several attempts were made to overcome those difficulties by describing the evolution in classical terms and solving large time problems by classical means. A lot of work was done in establishing rigorous

V.N. Alexandrov et al. (Eds.): ICCS 2001, LNCS 2073, pp. 447–452, 2001.
© Springer-Verlag Berlin Heidelberg 2001

relationship between classical and quantum tools for description of system evolution ([2]). Although formal mathematical solution of the problem was given in 1979 ([3]), and computational algorithm was proposed in 1986 ([4]), only recently it was possible to realize it on large systems ([5]). The main result of ([3]) was the functional transformation to so called interaction coordinates, that reduces Hamiltonian function of the problem to actually responsible for described process and reduces integration interval to finite one, corresponding to interaction region. More than that it transforms asymptotical boundary conditions to standard ones and makes it possible to get the expression directly to scattering amplitude as an average over Green's functions of the problem but in mixed representation and in interaction coordinates.

$$A(\boldsymbol{i}, \boldsymbol{f'}) = \langle \boldsymbol{G}(\boldsymbol{i}, \boldsymbol{t}) \boldsymbol{G}(\boldsymbol{t}, \boldsymbol{f'}) \rangle \ . \tag{1}$$

The average is taken over the coordinate space of the problem with Green's functions determined in terms of path integrals over the phase space with the weights of the type

$$\exp\left(-i + \int X dP + iXP - i \int H dt\right) \ . \tag{2}$$

i.e. the classical action in phase space in standard notation ([1]). Since we use Green's functions only for computation of averages (1) we can make any phase coordinates transformations of the type

$$H(P, X) \to H\left(X, \frac{\partial F}{\partial X}\right) + \frac{\partial F}{\partial t} \ . \tag{3}$$

With F being the Generator of the transformation. It is convenient to choose F as a solution of certain equation ([4]) that guarantees the possibility of evaluation of path integral with the weight (2). In that case instead of computation of path integral we have to solve four times the equation for F, that is more convenient since it is partial differential equation of the first order ([3]). The resulting amplitude representation is multidimensional integral:

$$\begin{aligned} T(P_i \to P_f) = \int dX_0 C \delta(X_0, P_0) \exp\left\{ \frac{i}{\hbar}(P_f X_f - P_i X_i) + \right. \\ \left. + \frac{i}{\hbar} F_1|_i^0 + \frac{i}{\hbar} F_1|_0^f + \frac{i}{\hbar} Q_i(Y_0 - Y_i) + \frac{i}{\hbar} Q_f(Y_f - Y_0) \right\} \ . \end{aligned} \tag{4}$$

2 Numerical Method

So in general case computation of scattering amplitude in our approach is reduced to computation of average, i.e. of integral over the coordinate space and solution for every coordinate of four partial differential equations of the first order for F. It is clear, that such formalism gives an ideal parallel algorithm, since we can do the solutions of equations for different points independently. More,

than that, we can make the same sort of diagonal approximation as in coupled states approach and choose one average solution for all four generators F. This is the equivalent of so called average trajectory approximation in semiclassical approach. Schematically the algorithm of such process computation can be presented in the following way ([6]):

I – Lagrangian surface construction for the system. The curvilinear coordinate system, within which all the further calculations are performed, is derived in it;

II – Classical trajectory problem solution. At this stage the system of ordinary non-linear differential equations of the first order is being solved numerically. The problem's parameters are collision energy E and quantum numbers of initial configuration n. This system is being solved by one-step method of 4th-5th order of accuracy. This method is conditionally stable (by initial deviation and right part), that's why the standard automatic step decreasing method is implied to provide its stability. It's worth mentioning that initial system degenerates in certain points. To eliminate this degeneration, the standard -procedure with differentiation parameter replacement is performed.

III – The results of classical trajectory problem calculation are used for quantum calculations performing and complete wave function obtaining in its final state. At this stage, the numerical problem represents solution of an ordinary non-linear differential equation of the second order. Calculating this equation is a difficult task due to non-trivial behavior of differentiation parameter ([7]). Differentiation algorithm consists of two stages: 1) construction of differentiation parameter values system using the results of classical problem calculation and 2) integration of initial differential equation on non-uniform system obtained by means of multi-step method. Choosing such integration step in a classical problem provides integration stability, while control is performed by means of step-by-step truncation error calculation. The obtained solution of differential equation is approximated in a final asymptote in a form of falling and reflected flat wave superposition;

Let's remind that calculations for steps II and III are made for specific values of collision energy E and oscillation quantum number of initial state. Results of these calculations allow to obtain one line of a transition matrix, which corresponds to $\boldsymbol{n}$. In order to obtain the entire transition matrix, calculations at stages II and III need to be repeated as many times as dictated by the size of transitional probability matrix. As a result the entire probability matrix is obtained. The procedure described needs to be repeated for many values of collision energy in order to enable further integration and velocity constants finding. It is clear, that most time consuming are the stages II and III and that they can be carried to large extend on independent computational systems, using one of them just to collect all the results and work out the statistics. Since from each of such computation we need only the value of the kernel of transition functional it was possible to make exchange of such information as low as possible. All the computations of the stages IV and V were carried out on MPP system Parsytec CCe-20 of IHPCDB and the individual computations for different trajectories on MPP IBM SP-2 of GMD. We found, that MPP architecture, although old-fashioned, is very

well suited for the architecture of proposed algorithm. The parallelization was performed for the values of collision energy. Calculation of classical trajectory problem, quantum calculation and transition probability matrix calculation is performed in each of the parallel branches. Let's note that just as in the case on non-parallelized algorithm all calculations from stages II and III are performed as many times as it is dictated by the size of transition probability matrix. Due to the fact that calculation in each of the parallel branches represents a separate problem and does not interact with other branches in calculation, the effectiveness of using this parallelization algorithm vs. relatively unparallelized algorithm is nearly proportional to a number of calculation branches, i.e. to the amount of computation nodes.

3 Numerical Example

As a reaction on which the algorithm was tested, a well studied bimolecular reaction Li + (FH) (LiFH)* (LiH) + H was taken. The potential surface for this reaction was reproduced using the quantum-mechanical calculations carried out in work ([8]). The results of testing have shown the calculation effectiveness to be nearly proportional to the number of computation nodes. We have proposed the variant of our approach for the shared memory systems. However now we have no technical possibilities to unite in large clusters systems with shared memory emulation in the regime of NUMA architecture. But this problem solution is one of the main items in the program of joint activities with GMD in the nearest years Finally we would like to stress one of the peculiarities of parallelization algorithms demonstrated - their scalability. Due to the fact that integration of transition probability matrix and rate constants calculation during stage V requires the values of matrix elements for many energy values, one can hardly find a supercomputer with an excessive number of computation nodes. As illustration (Fig. 1) we show first exact converging results of computation of reaction probability and properties of the system.

4 Conclusions

We have shown that the use of some physical considerations makes it possible to derive some new algorithms for solution of the evolution equations for physical variables like wave function. With those algorithms we can reduce the needed computer time orders of magnitude, go to substantially larger number of processor and work out approximate methods, which can be used for mass computations in technical applications. Scalability of these algorithms was used for conducting distributed computing runs on the supercomputers of GMD and SARA. Internet possibilities allowed to obtain access for most difficult part of the problem - trajectories calculations - to the far more powerful computing resources that are available in IHPCDB, and so to conduct distributed computations in the different regimes including X terminal regime. In the future we are planning to provide in the similar approach the visualization of the numerical

results, data preparations and preliminary tests for remote compilation on the cluster of workstations Octane and Sun Ultra, available in IHPCDB and sending them for further processing to the ONYX visualization supercomputers situated in GMD and SARA. At the same time the proposed approach really make it possible to get on large scale problems substantial speed-ups over the standard algorithms.

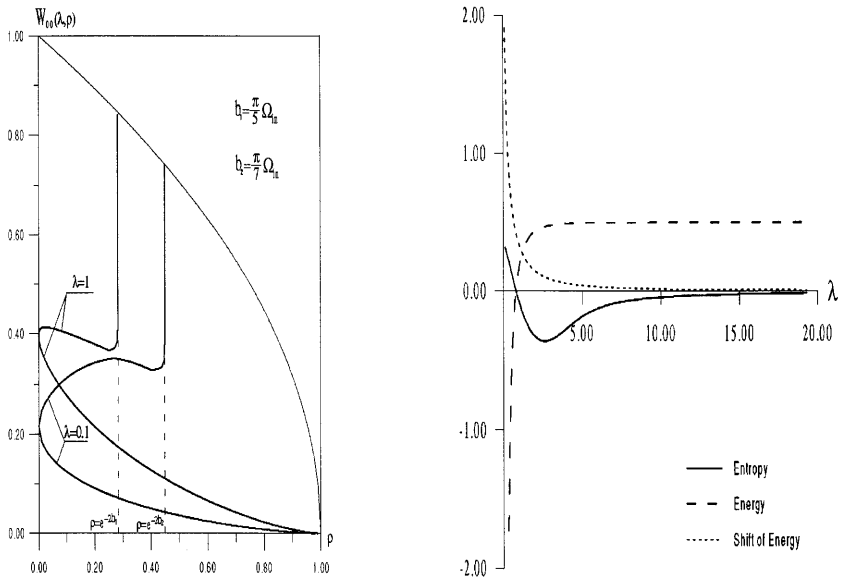


Fig. 1. The results of the first exact computation of the probability dependencies for reaction Li + (FH) (LiFH)* (LiH) + H

References

[1] Topaler M., Makri N.: Multidimensional path integral calculations with quasidiabatic propagators: Quantum dynamics of vibrational relaxation in linear hydrocarbon chains. J.Chem.Phys. Vol. 97, 12, (1992) 9001-9015

[2] Greenberg W.R., Klein A., Zlatev I.: From Heisenberg matrix mechanics to semiclassical quantization: Theory and first applications. Phys. Rev. A Vol.54, 3 , (1996) 1820-1836.

[3] Dubrovskiy G.V., Bogdanov A.V. Chem.Phys.Lett., Vol. 62, 1 (1979) 89-94.

[4] Bogdanov A.V.: Computation of the inelastic quantum scattering amplitude via the solution of classical dynamical problem. In:Russian Journal of Technical Physics, 7 (1986) 1409-1411.

[5] A.V. Bogdanov, A.S. Gevorkyan, A.G. Grigoryan, Stankova E.N.: Use of the Internet for Distributed Computing of Quantum Evolution. in Proccedings of 8th Int. Conference on High Performance Computing and Networking Europe (HPCN Europe '2000), Amsterdam, The Netherlands (2000)

[6] A.V. Bogdanov, A.S. Gevorkyan, A.G. Grigoryan and S.A. Matveev, Investigation of High-Performance Algorithms for Numerical Calculations of Evolution of Quantum Systems Based on Their Intrinsic Properties in Proccedings of 7th Int. Conference on High Performance Computing and Networking Europe (HPCN Europe '99), Amsterdam, The Netherlands, and April 12-14, 1999, pp.1286-1291.

[7] A.V. Bogdanov, A.S. Gevorkyan, A.G. Grigoryan, First principle calculations of quantum chaos in framework of random quantum reactive harmonic oscillator theory, in Proceedings of 6th Int. Conference on High Performance Computing and Networking Europe (HPCN Europe '98), Amsterdam, The Netherlands, April, 1998.

[8] S. Carter, J. N. Murrell, Analytical Potentials for Triatomic Molecules, Molecular Physics, v. 41, N. 3, pp. 567-581, (1980).

Complex Situations Simulation When Testing Intelligence System Knowledge Base

Yu.Nechaev, A.Degtyarev, A.Boukhanovsky

Institute for High Performance Computing and Data Bases, Fontanka 118, 198005
St.Petersburg,Russia
(int, deg, avb)@fn.csa.ru

Abstract. Construction of tool system for testing dynamic knowledge base of intelligence system (IS) of dynamic object (DO) behavior analysis and forecast is discussed. The system provides generation of environment and dynamics of floating DO - environment interaction in various operation conditions. Results of imitating experiment are given.

Introduction

Hardware and software design providing testing of knowledge base (KB) represents one of the important directions of the complex approach in development of intelligence technologies. At modern tool design the tendency of intellectualization in a direction of artificial intelligence methods application is precisely looked through. The analysis of existing tools show, that the main share of capacity and intellectuality of such toolkit is not associated with its architecture, but with functionalities of separate component of technological environment [1-4].

Packages APT KEE and G2 are among the most powerful and advanced tool systems. System G2 of firm Gensym is further development of system PICO and one of the most powerful environments for real-time systems. Due to openness of the interface and support of a wide spectrum of computing platforms system G2 allows to unite isolated automation means in a uniform complex control system. At the same time G2 is poorly adapted to perception of a complex information stream by development and testing of onboard integrated real-time systems. The tool system offered in the paper represents integrated component for the solution of a wide range of problems of ship's dynamics and offshore structures.

Novelty of the developed program technology consists in the following:

- the model of sea three-dimensional waves and wind in small-scale and synoptic ranges of variability is developed;
- the model of interaction of floating DO with an environment in various conditions of operation is developed;
- imitating experiment for DO dynamic characteristics assesment in extreme situations is carried out. This experiment is connected with loss of stability of oscillatory movement at a various level of external actions.

Principal difference of the tool system from other similar systems consists in simulation of a real picture of exerted actions. It provides reliability of the practical

V.N. Alexandrov et al. (Eds.): ICCS 2001, LNCS 2073, pp. 453–462, 2001.
©Springer-Verlag Berlin Heidelberg 2001

recommendations, which are given out by intelligence system in realization of inferential mechanism.

1. Architecture and functionalities of system.

The complex of the theoretical and practical questions connected with development of tools was considered in work [1]. Continuation of researches on perfection of the approach to system creation has allowed to formulate and realize technology of transforming the information when estimating DO behavior at any level of external actions.

The architecture and base components of the system are given in fig. 1.

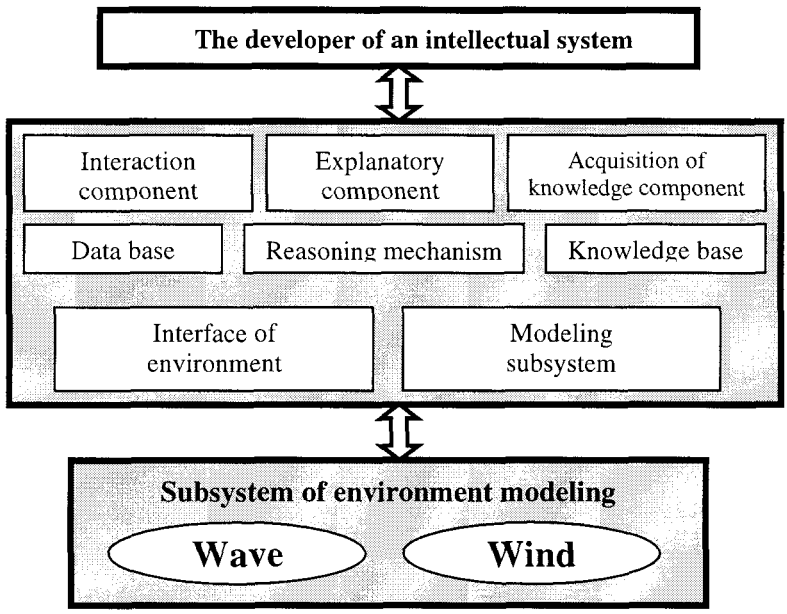


Fig. 1. Instrumental tool for knowledge base testing

The submitted structure includes traditional IS components (KB, database, solver, means of knowledge explanation and purchase) and the special complexes allowing the developer to interact with a subsystem of environment and object dynamics modeling. The basic principles fixed in a basis of system provide:

- technology of open systems, adaptability and dynamism;
- search of the decision based on fuzzy models of knowledge representation and imitating modeling;
- application of cognitive graphics and analysis of dynamic stages.

The testing problem is represented by the tuple:

$$<H_i, S_j, T_k>; \quad (1)$$

$$(i=1,\ldots,n;\ j=1,\ldots,m;\ k=1,\ldots,l)$$

where H, S, T are sets of hypotheses, symptoms and tasks.

Initial data (1) characterize matrixes:

$$G = \begin{array}{c} \\ S_1 \\ \vdots \\ S_m \end{array} \begin{array}{c} \begin{array}{ccc} H_1 & \cdots & H_n \end{array} \\ \begin{bmatrix} w_{11} & \cdots & w_{n1} \\ \vdots & w_{ij} & \vdots \\ w_{1m} & \cdots & w_{nm} \end{bmatrix} \\ \begin{array}{ccc} E_1 & \cdots & E_n \end{array} \end{array}; \quad R = \begin{array}{c} \\ T_1 \\ \vdots \\ T_\ell \end{array} \begin{array}{c} \begin{array}{ccc} & & \end{array} \\ \begin{bmatrix} v_{11} & \cdots & v_{1m} \\ \vdots & v_{kj} & \vdots \\ v_{\ell 1} & \cdots & v_{\ell m} \end{bmatrix} \\ \begin{array}{ccc} & & \end{array} \end{array} \begin{array}{c} \\ C_1 \\ \vdots \\ C_\ell \end{array} \tag{2}$$

Here E_i $(i=1, _, n)$ are estimations; C_k $(k=1, _, l)$ are expenses; w_{ij} $(i=1,\ldots,n;\ j=1,\ldots,m)$ is weight of a symptom in the given hypothesis; v_{kj} $(k=1,\ldots,l;\ j=1,\ldots,m)$ is characteristic vector specifying correspondence between a problem and symptoms.

In that specific case, at unequivocal correspondence between symptoms and problems matrix R is identity. Then the initial data can be described by replacement of appropriate tasks in matrix R and adding a vector-column of expenses on the right. Procedures of IS testing and diagnostics are based on application of acceptance decisions mechanisms with use of KB fuzzy models.

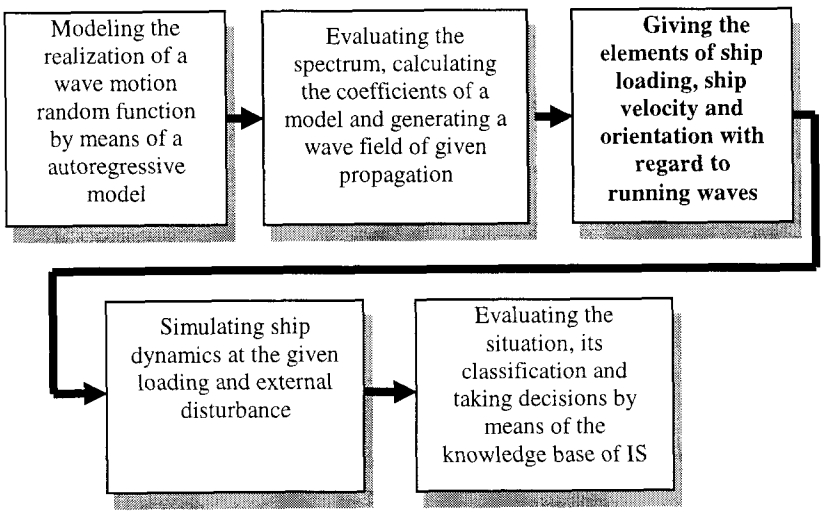


Fig. 2. Information flow in IS knowledge base using imitation modeling methods

2. Specific features of external actions modeling

The basic external actions influencing floating DO are determined by the irregular hydrodynamic forces of wind and wave character caused by surface wind, wind waves or swell, and also by fetch current. The integral property of these

hydrometeorological processes is their space and time domain variability caused by superposition and interaction of a large number of factors. Characteristic ranges of variability for floating DO operation are synoptic variability (time scale variation is from day up to several day) and small-scale fluctuations (cyclicity from several seconds up to one hour). Complexity, non-uniform scale, polycyclicity, great variability of hydrometeorological processes result in necessity to consider them as stochastic functions of time and spatial coordinates and to describe their properties in terms of probabilistic characteristics. Further we shall understand the probabilistic model as the concrete kind of stochastic process record. It allows to obtain data on any probabilistic characteristics and, at the same time, to take into account dependence of the process of the factors included in conditions complex. These models are necessary for generalization of results of the analysis, compression of the information, an establishment of relation between various probabilistic characteristics and reproduction of realizations in non-observable situations [5].

Presence of multiscale variability causes a non-stationarity of hydrometeorological processes, and a variety of influencing factors (stratification of surface atmosphere layer and sea water, movement of baric formations and synoptic whirlwinds at oceans and seas) results in spatial heterogeneity. It results in necessity of using complex of several parametrically related probabilistic models in two time ranges for tool system.

For reproduction of spatial-time wave surface field $\zeta(x, y, t)$ relative to average level it is allowable to use a hypothesis about a stationarity and uniformity of the initial field first formulated by M.S. Longuet-Higgins [6]. However, the spectral model offered by him has not found wide application in real time systems due to low speed of convergence and essential computing expenses. As alternative we used model of a field of autoregress in the form [7]

$$\zeta(x,y,t) = \sum_i \sum_j \sum_k \Phi_{ijk} \zeta(x-i, y-j, t-k) + \varepsilon(x,y,t) \qquad (3)$$

Here Φ_{ijk} is the autoregress coefficients describing spatial-time connectivity of a field $\zeta(x,y,t)$, $\varepsilon(x,y,t)$ is normal white noise with a dispersion σ^2, not dependent on (x,y,t).

Parameters Φ_{ijk} are determined by means of system of Yule-Walker equations through values of spatial-time correlation function $K_\zeta(x,y,t)$. However wave character of modeled surface motion (3) provides functional relation between spatial and time field components realizable as characteristics of the wave equation [8]. It allows to reduce regularity of initial problem setting model (3) by means of only spatial $S_\zeta(u,v)$ spectral density connected to correlation function by the ratio

$$K_\zeta(x,y,t) = \iint S_\zeta(u,v) \cos(ux + vy + \omega(u,v)t) du dv \qquad (4)$$

Here u, v are the wave numbers inversely proportional to wavelength in two orthogonal directions, $\omega=\omega(u,v)$ is the frequency determined by a dispersion relation kind of which depends on waveformation conditions. Alternative to (4) may be use of frequency - directed spectrum $S_\zeta(\omega,\Theta)$, connected with $S_\zeta(u,v)$ by means of nonlinear coordinates transformation [6].

Synoptic variability of waves is caused by formation, movement and disintegration of baric formations - cyclones and anticyclones accompanied with strong gradients of atmospheric pressure, and inducing alternation of storms and "weather windows". In

the work [9] for the description of wave fields variability in storm the concept of climatic spectra, as probabilistic characteristics of time rows of functions $S_\zeta(u,v,\bar{r},t)$ or $S_\zeta(\omega,\Theta,\bar{r},t)$, dependent on spatial coordinates $\bar{r}$ and time t is used. With a view of reduction of regularity let us consider a wave spectrum as the determined function of casual arguments Ξ describing waveformation conditions :

$$S(\omega,\Theta,\bar{r},t) = S(\omega,\Theta,\Xi). \quad \Xi = \Xi(\bar{r},t) \tag{5}$$

For example, in an affine vector Ξ the following parameters may be included: average wind waves h_w and swell h_s height, average wind waves τ_w and swell τ_s period, average directions of wind waves Θ_w and swell Θ_s propagation, and also parameters of form, for example, peakness parameter γ in JONSWAP approximation [10], also depending from wind speed $\bar{V}(\bar{r},t)$.

Thus, modeling of synoptic variability of spatial-time complex sea fields is reduced to reproduction of a casual affine vector field $\Xi(\bar{r},t) = \{h_w, h_s, \tau_w, \tau_s, \Theta_w, \Theta_s\}$. For this purpose in tool system the model of decomposition on orthogonal basis is used:

$$\Xi(x,y,t) = \sum_k \alpha_k(t)\Psi_k(x,y) \tag{6}$$

Here α_k (t) is the time-dependent scalar coefficients determining variability of field of parameters in time, $\Psi_k(x,y)$ are the affine basic functions (pair orthogonal). As optimum basis we use the natural orthogonal functions of an affine vector field determined as eigenfunctions of correlation nucleus through integral equation

$$\int K_\Xi(\bar{r}_1,\bar{r}_2)\Psi_k(\bar{r}_2)d\bar{r}_2 = \lambda_k\Psi_k(\bar{r}_1) \tag{7}$$

Here λ_k are eigennumbers of the equation; they characterize a variance of coefficients $\alpha_k(t)$.

Application of (6) for synoptic variability modeling of near-water wind speed is complicated by the fact that its volume $\bar{V} = (U,V)^T$ is a geometrical vector in Euclidian space: population mean $E[\bar{V}]$ is also a geometrical vector, and correlation function $K_V(t,\tau,\bar{r}_1,\bar{r}_2) = E[\bar{V}(t,\bar{r}_1)\otimes\bar{V}(\tau,\bar{r}_2)]$ is a second rank tensor characterized by a set of invariants. In view of it, decomposition (6) becomes

$$\bar{V}(x,y,t) = \sum_k \beta_k(t)\bar{\Psi}_k(x,y) \tag{8}$$

where $\beta_k(t)$ are scalar probabilistic processes, and $\bar{\Psi} = (\Psi_U,\Psi_V)^T$ are vector natural orthogonal functions defined as the solution of homogeneous integral equations system in dyadic form

$$\begin{aligned} &\int K_{UU}(\bar{r}_1,\bar{r}_2)\Psi_U(\bar{r}_2)d\bar{r}_2 + \int K_{UV}(\bar{r}_1,\bar{r}_2)\Psi_V(\bar{r}_2)d\bar{r}_2 = \lambda\Psi_U(\bar{r}_1) \\ &\int K_{VU}(\bar{r}_1,\bar{r}_2)\Psi_U(\bar{r}_2)d\bar{r}_2 + \int K_{VV}(\bar{r}_1,\bar{r}_2)\Psi_V(\bar{r}_2)d\bar{r}_2 = \lambda\Psi_V(\bar{r}_1) \end{aligned} \tag{9}$$

Here K_{UU}, K_{UV}, K_{VU}, K_{VV} are the appropriate components of correlation tensor. The eigenfunctions determined by this equation, as well as in (7), are orthogonal. They also provide the fastest convergence of decomposition (8) among all orthogonal bases with the square-law metrics.

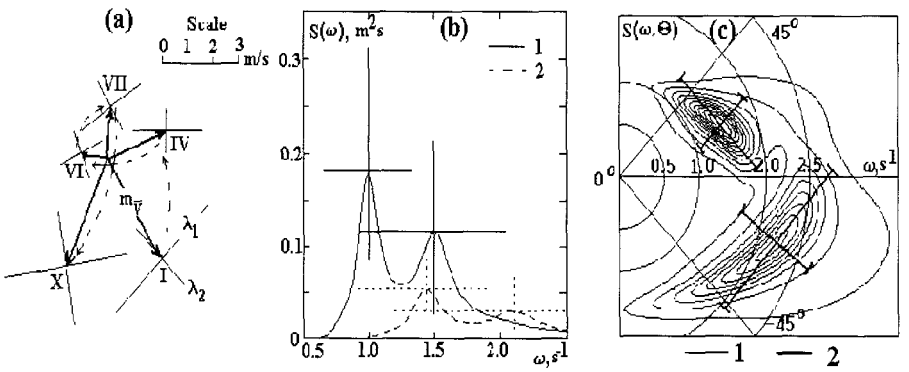
Application of natural orthogonal basis (7,9) in models (6,8) allows to take into account the heterogeneity of initial field caused, for example, by special features of waveformation conditions (differences of depths, irregularity of coastal line, etc.), complex relief of land, and also by presence of ice cover on a part of water area. Such approach provides transition from model of casual fields $\Xi(\bar{r},t)$, $\bar{V}(\bar{r},t)$ to model of time series $\{\alpha_k(t), \beta_k(t)\}$ of decomposition factors. In spite of the fact that for coefficients of decomposition on proper basis $\mathrm{cov}(\alpha_k(t_1),\alpha_j(t_2)) = 0$ and $\mathrm{cov}(\beta_k(t_1),\beta_j(t_2)) = 0$ at $t_1=t_2$ and $k\neq j$ are true, coefficients $\alpha_k(t)$ and $\beta_k(t)$ are correlated among themselves by virtue of functional dependence of wind and wave fields. For reproduction of system of related time series in tool system we use model of multivariate autoregression

$$A_t = \sum_k \Phi_k A_{t-k} + E_t \qquad (10)$$

As against model of a scalar field (3), here $A_t = \{\alpha_1(t),\ldots,\alpha_m(t), \beta_1(t),\ldots,\beta_n(t)\}^T$, $E_t=\{\varepsilon_1,\ldots,\varepsilon_{m+n}\}^T$ is the vector of correlated white noise, $\Phi_k=\{\phi_{ijk}\}$ are matrix coefficients of autoregression.

Taking into account, that it is possible to compare for each realization of variables $\{\Xi(\bar{r},t), \bar{V}(\bar{r},t)\}$ frequency- directed spectrum (5), the problem of wind wave fields modeling in view of its synoptic and small-scale variability is reduced to reproduction of spectrum parameters (6), wind speed (8) with use of (10), and the subsequent generation of sea surface elevation field relative to average level in model (3) identified from relation (4).

In the fig. 3 by the Barents Sea example some probabilistic characteristics of wave and wind used for identification and verification of probabilistic models are shown: average wind speed hodograph and the main axes variance tensor per months (a), frequency (b) and frequency- directed (c) climatic spectra of the complex sea with the tolerant intervals determining their synoptic variability. Also in fig. 3 reproduction results of wind and waves modeling fields in synoptic range of variability (d), and three-dimensional wave fields in vicinities of point (70N, 40E) are shown.



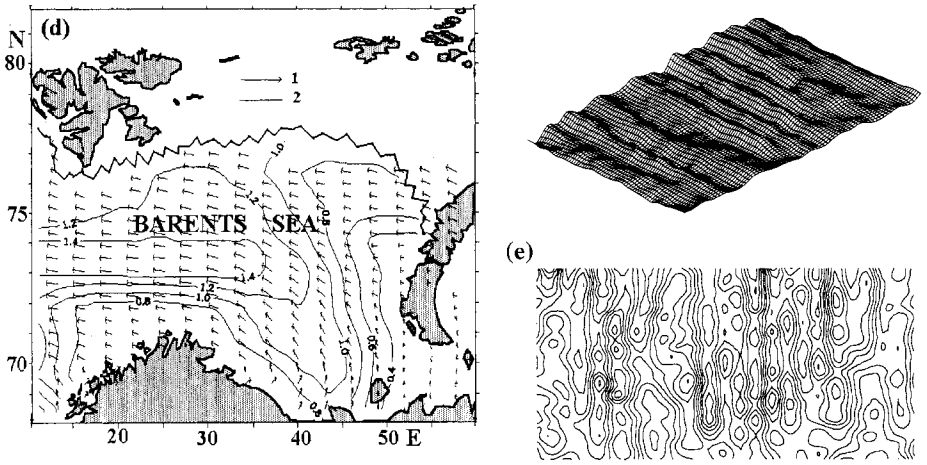


Fig. 3. Probabilistic characteristics of wind speed fields (a), wind waves (b,c), results of model calculations of wind and waves fields in synoptic range of variability (d) and in quasistationary range (e)

3. Special features of "DO-environment" dynamics interaction

Consideration of floating DO behavior under the wind and wave action is come to second Newton law:

$$\sum_i \vec{F}_i = m\bar{a}; \quad \sum_i \vec{M}_i = J\bar{\omega} \tag{11}$$

where $m\bar{a}, J\bar{\omega}$ are the principal vector and moment of considered dynamic system. Components in right part could be divided, in general, into disturbing, restoring, inertial and damping components.

Many authors represent (11) as model systems of differential equations like

$$F_i\left(x_i, \dot{x}_i, \ddot{x}_i, t, X_{i1}, \ldots, X_{im}, Y_{i1}, \ldots Y_{in}\right) = 0, \tag{12}$$

where $F_i(\bullet)$ are nonlinear functions; x_i are linear and angular variables; $X_{i1}, \ldots, X_{im}$ are the parameters describing DO as dynamic system (inertial, damping and restoring components); $Y_{i1}, \ldots, Y_{in}$ are exciting forces and the moments; t is time; i=1,2,...,6.

The experience shows that complication of model equations (12), terms improvement for disturbing actions and other components become ineffective at some moment. Terms' deriving is especially difficult in case of random actions, which arise from wind and waves.

Alternative to model equations is direct simulation of DO motion. Possibility of description of normal and transverse stresses distribution on DO wetted surface reduces simulation to clear procedure based on fundamental laws. It is necessary to

solve Navier-Stokes problem alongside the DO hull and only potential problem can be solved at some from the hull (beyond the boundary layer). One of the main obstacles in the problem solution is unknown boundary (wave surface) where boundary conditions are satisfied. But the methods described in the previous chapter permit simulate 3D random sea waves in any time range. The character of proposed methods is physically adequate. So generated wave fields are in accord with hydrodynamics.

In this case we can divide complex problem into two simpler problems:

1. generation of time-spatial wave fields near floating DO
2. simulation of velocity and pressure fields in non-viscous fluid.

This solution is initial approximation to wave diffraction problem and Navier-Stokes problem in boundary layer.

Solution of potential problem is reduced to the following equation in the bottom hemisphere

$$\Delta\varphi = 0$$

$$\frac{\partial\varphi}{\partial t} + \frac{1}{2}\left(\left(\frac{\partial\varphi}{\partial x}\right)^2 + \left(\frac{\partial\varphi}{\partial y}\right)^2 + \left(\frac{\partial\varphi}{\partial z}\right)^2\right) + g\zeta = p \quad \frac{\partial\zeta}{\partial t} + \frac{\partial\zeta}{\partial x}\frac{\partial\varphi}{\partial x} + \frac{\partial\zeta}{\partial y}\frac{\partial\varphi}{\partial y} = \frac{\partial\varphi}{\partial z} \tag{13}$$

This is the linear problem with nonlinear boundary conditions and unknown boundary. But as we mentioned above methods of external actions modeling permit to reduce this problem. So the problem with unknown boundary becomes to problem with known boundary at each time step when two above mentioned boundary conditions are fulfilled. In this case we know process ζ in any space point and at the any time, and we can calculate any derivations in space and time domain using additional assumptions about wave nature (local wave numbers) [12]. Such way permits to obtain potential derivations on the wave surface and we can calculate them in any point in bottom hemisphere with the help of Newton potential theory.

Calculation of hydrodynamic pressure by means of Bernoulli's integral enables to calculate hydrodynamic forces and moments acting at the considered time moment.

$$\vec{F} = -\int_{S_0} p \cdot \vec{n} dS \qquad \vec{M} = -\int_{S_0} p \cdot (\vec{r} \times \vec{n}) dS \tag{14}$$

where S_0 is the wetted ship surface, n is a radius - vector of points of the wetted DO surface.

So the algorithm of DO motion simulation at sea could be proposed in the following form [11]:

1. Consequence of wave fields in considered region are generated $\zeta(x_i,y_j,t_k)$, where i=1.....N; j=1..M (N×M – considered region), $t_k=k\cdot\Delta t$, k=0..L (considered time moments)
2. All necessary wave characteristics are calculated.
3. Let us take any initial conditions for integration beginning. (At first moment we can assume that system pole coincides with centre of gravity)

The following items are fulfilled for each time step

4. The cross-points of wave and ship hull are found.
5. Pressure is calculated at wetted hull points.
6. Exiting force and moment components are calculated.

7. Restoring forces and moments are calculated
   Submerged volume is calculated by integration.
   Central of buoyancy is calculated too.
8. The system of all forces and moment is reduced to one main force and main moment (at first moment exciting force acts in the centre of gravity)
9. Let us integrate the system of differential equations at one time step. In this case such system has the following form that obtained from second Newton law:

$$D \cdot \ddot{X} + F_d(\dot{X}) = F \qquad (15)$$
$$J \cdot \ddot{\Phi} + M_d(\dot{\Phi}) = M$$

where D is DO mass, X is a linear displacement (surging, swaying or heaving), J is a correspondent DO inertia moment, Φ is an angle displacement (rolling, pitching, yawing), F_d and M_d are the correspondent damping forces and moments, F and M are the components of main force and main moment.

10. Momentary centre of motion is obtained. This is the point of excitation forces action at the next time step.

4. Results of experiments

The IS full-scale tests have been carried out aboard a tanker in the Baltic Sea, on a container ship in a voyage in the Mediterranean and in the Atlantic, on a small ship in the Black Sea.

The test proved the possibility of practical evaluation and prediction of the dynamic characteristics, wind waves, the reliability of KB operation under various conditions of service. KB was design in accordance with preliminary situation simulation. For these purposes hydrodynamic and probabilistic modeling of wind waves and ship motion were fulfilled.

The table shows the comparative data about the different ways of measuring seas parameters obtained at the containership. In the table you can see that the IS method applied to the seas estimation on the basis of identification method gives good results which correspond to the actual measurement data received by means of standard wave-meters and systems (string wave recorder, laser sensor, wave recorder GM-32).

Table 1. Wave height measurement (3% quantile, m)

Wave force (balls)	Laser sensor	Wave recorder	String wave recorder GM-32	Designed algorithm	Max Div. %,
3	-	-	1,17	1,23	5,1
4	1,80	1,74	1,76	1,85	6,3
5	3,28	3,17	-	3,40	7,2
6	5,72	5,61	-	5,83	3,9

Acknowledgement

The work is supported by grants INTAS Open 1999 - N666 and RFBR 00-07-90227

References

1. Nechaev Yu.I., Degtyarev A.B. Account of peculiarities of ship's non-linear dynamics in seaworthiness estimation in real-time intelligence systems. Proc. of the 7th International Conference on Stability of Ships and Ocean Vehicles. Launceston, Tasmania, Australia, February 2000, vol.B, pp.688-701.
2. Nechaev Yu.I., Degtyarev A.B., Boukhanovsky A.V. Analysis of extremal situations and ship dynamics in seaway in intelligent system of ship safety monitoring. Proc. of the 6th International Conference on Stability of Ships and Ocean Vehicles. Varna, Bulgaria, September 1997, vol.I, pp.351-359.
3. Boukhanovsky A.V., Degtyarev A.B. The instrumental tool of wave generation modelling in ship-borne intelligence systems. Proc. of 3rd International Conference CRF-96, St.Petersburg, Russia, June 1996, vol.I, pp.464-469
4. Nechaev Yu.I., Degtyarev A.B., Boukhanovsky A.V. Adaptive forecast in real-time intelligence systems. Proc. of 13th International conference on hydrodynamics in ship design HYDRONAV'99, Gdansk-Osroda, Poland, September, 1999, pp.229-235.
5. Lopatoukhin L.J., Rozhkov V.A., Bukhanovsky A.V. The main problems of wind and wave statistics, based on spectral modelling data. Proc. of Coastal Wave Meeting, September, 25–28, Barselona, Spain, 2000, paper 7.5.
6. Longuet-Higgins M.S. The statistical analysis of a random moving surface. *Phil. Trans. Roy. Soc.,* London, 1957, **249**, N966, pp.321-387
7. Rozhkov V.A., Trapeznikov Yu.A. Probability models of oceanological processes. – Leningrad, Gidrometeoizdat P.H., 1990 (in Russian)
8. Komen G.L., Cavaleri L., Donelan M., Hasselmann K., Hasselmann S., Janssen P. Dynamics and modelling of ocean waves. Cambridge University Press., 1994.
9. Boukhanovsky A.V., Lopatoukhin L.J., Rozhkov V.A. Wave climate spectra and wave energy resources in some Russian seas. WMO/TD - No 938 "Provision and engineering/operational application of ocean wave data", 1998, pp.324-333.
10. Hasselmann K. et al. Measurements of wind-waves growth and swell decay during the Joint North Sea Wave Project (JONSWAP). – Hamburg: Deutsch. Hydrogr. Inst., 1973.
11. Degtyarev A.B., Podolyakin A.B., Imitative modelling of ship behaviour in random sea. Proc. of International shipbuilding conference'98, St.Petersburg, Russia, November 1998, vol.B, pp.418-426. (in Russian)
12. Whitham G.B. Linear and nonlinear waves. – New York: John Wiley & Sons, 1974.

Peculiarities of Computer Simulation and Statistical Representation of Time–Spatial Metocean Fields

A. Boukhanovsky[1], V. Rozhkov[2], and A. Degtyarev[1]

[1]Institute for high performance computing and data bases
Fontanka 118, 198005 St.Petersburg, Russia
[2]St.Petersburg branch of State Oceanographic Institute
23 linia 2A, 199026 St.Petersburg, Russia
avb@fn.csa.ru, ocean@nwmet.nw.ru, deg@fn.csa.ru

1. Introduction

The integral property of hydrometeorological fields (atmospheric pressure, wind speed, wind waves, temperature and salinity of seawater, sea currents) is their spatial–time variability caused by superposition and interaction of a large number of factors. Characteristic ranges of variability are interannual variability (cyclicity of fluctuations more than one year), annual cycles, synoptic variability (time scale of fluctuations from one day to several days), daily cycles, small-scale fluctuations (cyclicity from several seconds to one hour). The presence of multiscale variability causes nonstationarity of hydrometeorological processes. Variety of active factors results in spatial heterogeneity of fields (stratification of surface layer and seawater, movement of baric formations in an atmosphere and mesoscale eddies in oceans and seas).

Traditionally the basis of objective laws of description of hydrometeorological field's variability is the analysis of full-scale data: shipboard observations during voyages, continuous observations at sea or coastal stations, satellite information. The intensive development of mathematical modeling methods on the basis of analytical and numerical solutions of system of termo- and hydrodynamics equations under the appropriate initial and boundary conditions has allowed to fill up considerably this information base. It results from both reproduction of the measurement data in points of a regular grid by means of reanalysis [12], and obtaining information about non-observable parameters on verified models. As a result, for example, we have wind waves [13], termohaline structure of waters and ecological system parameters [9].

The specificity of application of hydrodynamic models for observation data assimilation in information base consists in obtaining results as a file of values $\Xi = \{\xi_p\}_{p=1}^{m}$ in different spatial points $\vec{r}(x_i, y_j, z_k)$ at the time moment t_s. Complexity, non-uniform scale, polycyclicity and great variability of the hydrometeorological information result in necessity to consider them as stochastic functions of time and spatial coordinates and to describe their properties in terms of probabilistic characteristics [16]. There are mean value $m_\Xi(\vec{r},t)$, variance $D(\vec{r},t)$, covariation function $K_\Xi(\vec{r},\vec{\rho},t,\tau)$ and spectral density $S_\Xi(\omega,\vec{r},t)$, which depend on several variable (coordinates r and time t, frequency ω, spatial $\vec{\rho}$ and time τ shifts). Traditional problem of multivariate statistical analysis (MSA) is estimation of such characteristics on natural data.

V.N. Alexandrov et al. (Eds.): ICCS 2001, LNCS 2073, pp. 463-472, 2001.
© Springer-Verlag Berlin Heidelberg 2001

Classical MSA operates with concepts of multivariate stochastic value, system of dependent stochastic values and multivariate time series [2,4]. The purpose of the paper the use of MSA methods with regard for specificity of spatial-time fields obtained by hydrodynamic simulation.

2. Hydrodynamic Simulation as Metocean Data Source

The evolution of surface wave field in space and time is governed by the basic transport or energy balance equation [17]

$$\frac{\partial S}{\partial t} + v \cdot \nabla S = G \tag{1}$$

where $S(\omega,\theta,\vec{r},t)$ is the two-dimensional wave spectrum, dependent on frequency ω and propagation direction θ; $v=v(\omega,\theta)$ is the group velocity. S is net source function. It is represented as the sum of the input S_{in} by the wind, the nonlinear transfer S_{nl} by resonant wave-wave interaction, and the dissipation S_{ds}. There are some other terms (interaction with slowly variable currents, etc.) which are normally small. They are not included in the propagation operator.

Equation (1) describes functional relation between fields of atmosphere pressure, wind and waves. There are many calculation models based on (1) devoted to obtaining time-spatial wave field. All they are differ from one another by sources function presentation and computational layout.

The first wave model which was realized as world famous software is WAM-model [17]. The theory and methods of numerical simulation are continuously improved. Now we have new results and models (WAVEWATCH [18], PHIDIAS [19], TOMAWAC [20], INTERPOL [21]) for deep and (SWAN [22]) for shallow water. Such results and a great activity in the field of reanalysis of pressure and wind in points of regular mesh [12,13] give possibility to use results of numerical simulation of time-spatial fields as initial data for analysis by means of MSA.

Specific character of computer presentation of hydrometeorological fields information is large volume of used data and long time for calculation. Hence, application of high performance computers is necessary. Part of the results used in this paper for MSA was obtained in Institute for High Performance Computing on supercomputer HP SPP1600 (8 processors).

3. Metocean Events as Elements of Functional Spaces

The basic probability model for the analysis of the hydrometeorological data is the random function $\eta(\vec{r},t)$ of spatial coordinates $\vec{r}$ and time t characterized by mean value

$$m_\eta(\vec{r},t) = E[\eta(\vec{r},t,w)] \tag{2}$$

and covariance function

$$K_\eta(\vec{r},\vec{\rho},t,\tau) = E[\eta^0(\vec{r},t,w)\cdot\eta^0(\vec{r}+\vec{\rho},t+\tau,w)], \tag{3}$$

where $E[\bullet]$ is the operator of population mean (averaging on ensemble of realizations numbered by index w), $\eta^0(\bullet)=\eta(\bullet)-m_\eta(\bullet)$.

By example of the three interconnected fields: atmospheric pressure p($\vec{r}$,t), wind speed $\vec{V}$ ($\vec{r}$,t) and wind waves S(ω,Θ,$\vec{r}$,t). It is easy to make sure that the operations of addition and multiplication in (1) and (2) are subject to a concrete definition owing to specificity of fields p(•),$\vec{V}$ (•),S(•). For each value ($\vec{r}$,*t*) the field of pressure p(•) is scalar value, the field is vectorial value depending on a gradient of scalar field p(•), field of wind waves computed in accordance with field $\vec{V}$ (•) through the equation of wave energy balance (1) is represented by frequency directional spectrum S(ω,Θ,•), where ω is frequency, Θ is mean direction of waves propagation.

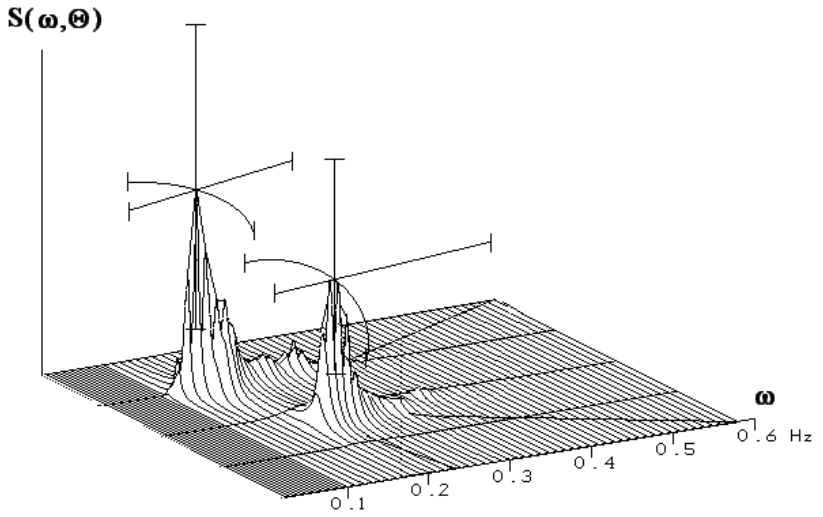


Fig. 1. Frequency-directed climatic spectrum of complex sea. North-Eastern part of the Black Sea

Hence, expressions (2) and (3) do not require comments only for p(•). Addition operation in (2) is carried out in accordance with "parallelogram rule" for $\vec{V}$ (•). Multiplication operation in (3) is understood as tensor product. Then $m_{\vec{V}}$ (•) is vector, and $K_{\vec{V}}(\bullet)$ is dyadic tensor [15]. Spectrum S(ω,Θ,$\vec{r}$,t) is function for wave field. Therefore interpretation of $m_s(\bullet)$ and $K_s(\bullet)$ is obvious only when (ω,Θ) are fixed.

$$S(\omega,\Theta) = S(\omega,\Theta,\Xi) \qquad (4)$$

where Ξ is parameters set ξ_j, $j=\overline{1,k}$, forming affine vector. Spectral moments or average values associated with them are used as parameters of observed waves elements. Relations for mean vector, its variance and specific quantiles can be obtained with the help of statistical linearization method. Approach (4) issues from the given function of joint distribution $F_\Xi(z_1,...,z_k) = P\{\xi_1 < z_1,...,\xi_k < z_k\}$ of systems of parameters Ξ. In fig. 1

the example of calculation of an average spectrum $S(\omega,\Theta,\overline{\Xi})$ of complex waves (Northeast part of the Black Sea), and also correspondent probabilistic intervals is shown.

In table 1 characteristics of three functional spaces typical for hydrometeorological fields are shown (scalar, Euclidean vector and affine vector).

Table 1. Functional spaces for hydrometeorological fields description

Object	Population mean	Variation	Scalar product	Decomposition	Examples
Scalar ζ	$E[\zeta]=\int \zeta d\mu$	$D[\zeta]=$ $E[(\zeta^0)^2]$	$(\zeta,\eta)=$ $\int \zeta\eta d\vec{r}dt$	$\sum_k a_k\varphi_k(\vec{r},t)$	atmosphere pressure field, and air temperature
Euclidean vector	$E[\vec{V}]=\begin{bmatrix}E[u]\\E[v]\end{bmatrix}$	$D[\vec{V}]=$ $E[\vec{V}^0\otimes\vec{V}^0]$	$(\vec{V}_1,\vec{V}_2)=$ $\int u_1u_2d\vec{r}dt+$ $\int v_1v_2d\vec{r}dt$	$\sum_k a_k\vec{\Psi}_k(\vec{r},t)$	wind speed field, sea currents.
Affine vector	$E[\Xi]=\begin{bmatrix}E[\xi_1]\\\cdots\\E[\xi_n]\end{bmatrix}$	$D[\Xi]=$ $E[\Xi\cdot\Xi^T]$	$(\Xi,H)=$ $\int Tr(\Xi\cdot H^T)d\vec{r}dt$	$\sum_k a_k\begin{bmatrix}\Phi_{1k}(\vec{r},t)\\\cdots\\\Phi_{nk}(\vec{r},t)\end{bmatrix}$	wave parameters field, temperature, salinity and oxygen in water

Note: $d\mu=\mu'(\zeta)d\zeta$ is a measure defined by density of uninterrupted distribution of scalar ζ.

Operations of addition in (2) and multiplication in (3) define rules of actions with ensemble elements. For the further simplification of model of the analysis let us introduce rules of operations with elements of space $\vec{\mathbf{r}}$ and time t, by means of scalar product. From table 1 it is clear, that only concept of scalar product is obvious to scalar values. For Euclidean and affine vectors it generalizes concept of scalar product both in discrete space and in continuous space.

Spaces with scalar product defined in table 1 will be Hilbert [3]. Hence, in each of them any element can be presented as an infinite converging series on some system of basic elements of this space: scalar functions $\varphi_k(\vec{r},t)$, Euclidean $\vec{\Psi}_k(\vec{r},t)$ or affine $\Phi_k(\vec{r},t)$ vectors - functions. Let us use this decomposition as fundamental way for model of statistical analysis simplification and reduction of dimension. They allow to proceed from scalar or vector function to an accounting set of scalar coefficients, i.e. to replace model of stochastic function by system of random values.

4. Statistical Representation of Time–Spatial Metocean Fields

Decomposition on basic elements in finite dimensional space is one of classical MSA procedures [2]. Decomposition coefficients a_k are called canonical variable. The principal components explaining general variability, factor loadings determining correlation structure, and canonical correlations representing degree of interrelation between two objects are considered depending on the purposes decomposition. The problem of obtaining canonical variables in the analysis of continuous hydrometeorological fields $\eta(\vec{r}, t)$ was traditionally solved by their representation as system of random values $H=\{\eta_1,\ldots,\eta_n\}$ in characteristic points $\{\vec{r}_i\}$ or/and at time

moments $\{t_i\}$, $i=\overline{1,n}$ with correlation matrix $K_H=E[H\cdot H^T]$. Classical procedures of matrix algebra were applied for transition to canonical basis [5].

In fig. 2 as an example spatial and time correlation functions of surface atmospheric pressure field are shown. From the figure it is clear, that they decrease slowly enough; the degree of coherence between rather far points is high. Hence, it is possible to speak about quasi-homogeneous areas. In this case using of η_i,η_j values in H_n in close points $(\vec{r}_i,\vec{r}_j)$ results in multicollinearity and estimation of correlation matrix K_H appears poorly worded and numerically singular. In generalizing work [1] it is noted that now neither quantitative criteria of multicollinearity, nor universal methods of its elimination by means of matrix algebra exist.

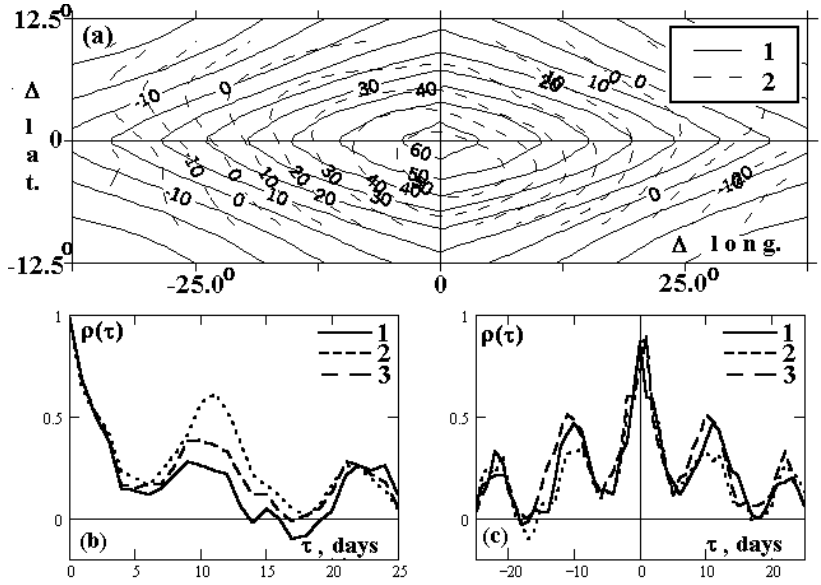


Fig. 2. Estimations of spatial and time correlation functions of pressure field. a) – spatial covariation function $K_p(x,y,\tau)$ (hPa^2) of pressure field over the Barents Sea in homogeneous approximation: 1–τ=0, 2–τ=24h. (b) –autocorrelation functions of pressure in characteristic points over the Barents Sea. 1– (75^0N, 30^0E), 2– (75^0N,50^0E), 3– (70^0N, 40^0E).(c) – joint correlation functions of pressure in characteristic points over the Barents Sea. 1–(75^0N, 30^0E) and (75^0N,50^0E), 2–(75^0N,30^0E) and (70^0N,40^0E), 3–(75^0N,50^0E) and (70^0N, 40^0E).

Therefore for canonical variable introduction let us resort directly to methods of decomposition in functional spaces shown in table 1. Karhunen [11] and Loeve [14] show that for scalar stochastic functions statistically orthogonal basis is generated by the homogeneous Fredholm integrated equation with the symmetric positive definite kernel

$$\int_{\langle r\rangle} K(\vec{r},\vec{r}_1,t,\tau)\varphi(\vec{r}_1,\tau)d\vec{r}_1 d\tau=\lambda\varphi(\vec{r},t) \qquad (5)$$

Spectrum of such kernel is discrete. Application of quadrature methods for (4) solution results in matrix representation without avoiding multicollinearity of the

problem. Therefore let us use projective (variational) methods [6] for obtaining orthogonal basis. It improves conditionality of the problem due to new appropriate orthogonal basis. Occasionally such method allows to obtain analytical solution for some types of modeling representations of autocorrelation function $K_\eta(\bullet)$ of non-homogeneous field. For example, one of the elementary models describes non-homogeneous field (in accordance with variation $\sigma^2(t)$) with correlation function of $K_\zeta(t,s)=\sigma(t)\sigma(s)k(t-s),\quad t,s\in[-1,1]$ .kind. Let us assume that $\sigma(\bullet)=\sigma_0+\beta t$ and $k(t-s)=1-|t-s|/2$. Then asymptotic expressions for the first two eigenvalues from (5) look as follow:

$$\lambda_1=\frac{1}{105}\left[27\beta^2+91\sigma_0^2+\sqrt{Q}\right],\quad \lambda_2=\frac{1}{105}\left[27\beta^2+91\sigma_0^2-\sqrt{Q}\right], \tag{6}$$
$$Q=169\beta^4+8134\beta^2\sigma_0^2+2401\sigma_0^4.$$

As an example, values λ_1, λ_2, obtained with the help of quadrature matrix procedure at the given number of knots N of a uniform grid, and by means of (5) for various combinations (σ_0,β), are shown in tab. 2. From the tab. we can see that convergence of quadrature (matrix) method is rather slow. By virtue of specificity of substitution of a numerable spectrum by a finite one, such spectrum produces upper estimate for λ_1^*,λ_2^* monotonously converging to true value. Using of analytical solution (5) in all cases gives close enough results, especially for λ_1^*. Estimations of λ_2^* are more different from those obtained by matrix method; their specification requires increase order of asymptotic decomposition.

Table 2. Comparison of convergence for matrix (quadrature) method and analytical approximation (6) by variational method.

Matrix method		σ_0=1,β=0.1	σ_0=1,β=0.3	σ_0=1,β=0.5	σ_0=1,β=0.7
N	5	1.52 0.59	1.66 0.55	1.90 0.50	2.22 0.48
	10	1.44 0.49	1.54 0.46	1.72 0.42	1.97 0.40
	100	1.37 0.41	1.45 0.39	1.59 0.36	1.78 0.34
	500	1.36 0.40	1.44 0.38	1.58 0.36	1.77 0.33
Analytical approximation (6)		1.34 0.40	1.42 0.36	1.57 0.30	1.76 0.23

Each eigenvalue λ_i sets variance of i-th principal component. This is coefficient a_i, which is determined by inverse transformation of tab. 1

$$a_k=\int_{\langle r\rangle}\zeta(\vec{r},t)\varphi_k(\vec{r},t)d\vec{r}dt \tag{7}$$

For vector random field (wind speed or sea currents) $\vec{V}(\vec{r},t)$ from tab. 1 representation of orthogonal basis at transition to the principal components is ambiguous. Let us consider the problem of obtaining statistically orthogonal basis for

a random vector field $\vec{V}(\vec{r})=(u,v)$. In this case decomposition coefficients a_k are not correlated, and the vector basis is the solution of system of homogeneous Fredholm equations

$$\begin{aligned}\int K_{uu}(\vec{r}_1,\vec{r}_2)\varphi(\vec{r}_2)d\vec{r}_2+\int K_{uv}(\vec{r}_1,\vec{r}_2)\psi(\vec{r}_2)d\vec{r}_2=\lambda\varphi(\vec{r}_1),\\ \int K_{vu}(\vec{r}_1,\vec{r}_2)\varphi(\vec{r}_2)d\vec{r}_2+\int K_{vv}(\vec{r}_1,\vec{r}_2)\psi(\vec{r}_2)d\vec{r}_2=\lambda\psi(\vec{r}_1),\end{aligned} \quad (8)$$

with respect to components $\vec{\Psi}=(\varphi,\psi)$. Orthogonal basis generated by (8) defines operator transformation $\Psi=Q\Phi$ resulting correlative tensor

$$K_{\vec{V}}(\vec{r}_k,\vec{r}_p)=\sum_k\sum_j\lambda_{ij}\left[\vec{\Psi}_i(\vec{r}_k)\otimes\vec{\Psi}_j(\vec{r}_p)\right] \quad (9)$$

in diagonal form:

$$\Psi K\Psi^T=Q\Phi K\Phi^T Q^T=QK'Q^T=\Lambda \quad (10)$$

where Λ is diagonal tensor, composed of variations of decomposition coefficients a_k. Inner orthogonal transformation Φ from tensor K to K' defines turn of principle basis $(\vec{e}_1',\vec{e}_2')$ of vector space relative to natural basis $(\vec{e}_1,\vec{e}_2)$ in each point $\vec{r}_k$. Tensor-function K' can be presented as

$$K'_{kp}(\vec{r}_k,\vec{r}_p)=\begin{bmatrix}\lambda_1(\vec{r}_k,\vec{r}_p) & \vartheta(\vec{r}_k,\vec{r}_p)\\ -\vartheta(\vec{r}_k,\vec{r}_p) & \lambda_2(\vec{r}_k,\vec{r}_p)\end{bmatrix} \quad (11)$$

where λ_1, λ_2 are principle axes of covariation tensor between points $(\vec{r}_k,\vec{r}_p)$, ϑ is indicator of rotation (when k=p ϑ is equal zero). Transit to principle basis $(\vec{e}_1',\vec{e}_2')$ for each pair $(\vec{r}_k,\vec{r}_p)$ allows to consider components of vectors $\vec{V}=(u,v)$ independently. Therefore outer orthogonal transformation Q of tensor K' to Λ defines rotation of sample axis in fundamental space over area $\langle r\rangle$.

For an explanation of correlation structure of scalar and vector random fields, techniques of factor analysis are used. This is representation of random field $\eta(\vec{r},t)$ as decomposition on the limited number m of coefficients a_k [8], specifying correlation function $K_\eta(\vec{r}_1,\vec{r}_2,t,\tau)$ as

$$K_\eta(\vec{r}_1,\vec{r}_2,t,\tau)=\sum_{i=1}^{m}\lambda_i\varphi_i(\vec{r}_1,t)\varphi_i(\vec{r}_2,\tau)+k_\varepsilon(t,\tau) \quad (12)$$

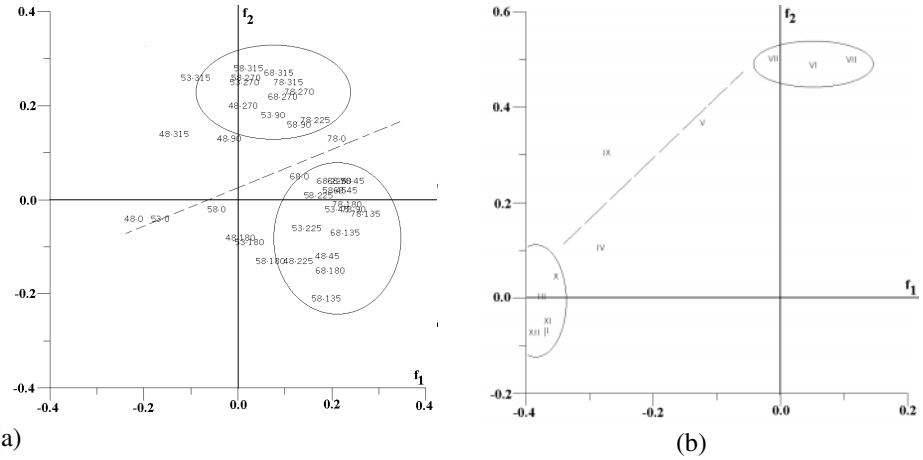
Here $k_\varepsilon(t,\tau)$ is correlation function describing variability of specific and random factors $\varepsilon(t)$. For scalar random fields basis functions (factor loadings) are defined mostly with the help of principal factors method. It consists in application of equation (5) to correlation function corrected on value $k_\varepsilon(t,\tau)$ [7].

In hydrometeorology for description of spatial and time connectedness two techniques of factor analysis connected with methods of correlation function

construction $\bullet_\eta(\vec{r}_1,\vec{r}_2,t,\tau)$ are traditionally distinguished. The first is the S-technique, when sections $K_\eta(\vec{r}_1,\vec{r}_2) = M[\eta^0(\vec{r}_1,t)\eta^0(\vec{r}_2,t)]_t$ of spatial correlation function are considered. Numeration of ensemble elements is defined by time *t*. Otherwise, when $K_\eta(t,\tau) = M[\eta^0(\vec{r},t)\eta^0(\vec{r},\tau)]_{\vec{r}}$, we speak about *T*-technique which explains time connectedness.

As an example let us consider double factor model of average monthly variability of surface level atmosphere pressure field in the North hemisphere, which is constructed with the help of S– and T– techniques. In fig.3 structure of factor loadings is presented for these two techniques.

Within the framework of S-technique the first factor (coefficient a_1, axis f_1) explains 41%, and defines the influence of processes occurring over the Euroasian continent and the Pacific Ocean. The second factor a_2 (20 % of variability, axis f_2) defines processes over the North American continent and Western part of the Atlantic Ocean. Western part of the Atlantic region influences both factors approximately equally.



(a) (b)

Fig. 3. Graphic representation of double factor model of correlation structure of atmosphere pressure field over North hemisphere (a) – S–technics: each point is denoted as LATITUDE (N)_ LONGITUDE (E) (b) T–technics: each month is denoted by serial number

Within the framework of T-technique in accordance with many years data in the North hemisphere the first factor explains 51% of variability and characterizes intensity of processes in autumn-winter season (October-February). The second factor (20% variability) characterizes summer season (June-August). The range corresponding to interseason defines both factors equally.

For description of spatial and time connectedness of system of two dependent random fields $\eta(\vec{r},t)$, $\zeta(\vec{r},t)$ or time connectedness of two spatial areas $\eta(\vec{r}_1,t)$, $\eta(\vec{r}_2,t)$ canonical correlation analysis is used [10]. It allows to explain not only structure of auto correlation functions $K_\zeta(\bullet)$, $K_\eta(\bullet)$, but also joint correlation function $K_{\zeta\eta}(\bullet)$. For this purpose let us introduce in appropriate functional spaces joint canonical basis functions $c_k(\vec{r},t)$ and $d_k(\vec{r},t)$ and appropriate canonical variables

$$U_k(t)=\int_{\langle r\rangle}\eta(\vec{r},t)c_k(\vec{r},t)d\vec{r},\quad W_k(t)=\int_{\langle r\rangle}\zeta(\vec{r},t)d_k(\vec{r},t)d\vec{r} \tag{13}$$

for which value of joint correlation function is

$$\rho(U_k(t),W_k(t+\tau))\to max,\quad E[U^2]=E[W^2]=1. \tag{14}$$

for any $\tau\geq 0$. Solution of (13–14) as the problem of conditional optimization in functional space results in system of two uniform integral Fredholm equations:

$$\begin{cases}-\lambda\int_{\langle r\rangle}K_{\eta\eta}(\vec{r},\vec{r}_1)c(\vec{r}_1)d\vec{r}_1+\int_{\langle r\rangle}K_{\zeta\eta}(\vec{r},\vec{r}_1)d(\vec{r}_1)d\vec{r}_1=0,\\ \int_{\langle r\rangle}K_{\zeta\eta}(\vec{r},\vec{r}_1)c(\vec{r}_1)d\vec{r}_1-\lambda\int_{\langle r\rangle}K_{\zeta\zeta}(\vec{r},\vec{r}_1)d(\vec{r}_1)d\vec{r}_1=0,\end{cases} \tag{15}$$

relative to functions $\bullet(\vec{r})$, $d(\vec{r})$ for stipulated τ and eigenvalues λ. By example of atmospheric pressure field in synoptic range of variability in tab. 3 values of first $\lambda_1(\tau)$ and second $\lambda_2(\tau)$ functions of canonical correlation for two spatial areas (Northwest of Atlantic and Europe) are given. From the table it is clear, that the functions $\lambda_i(\tau)$ decrease slowly.

Table 3. Canonical correlation functions for the Northeast Atlantic and Europe. Synoptic variability. Autumn season (October-November).

τ, days	0	5	10	15	20
$\lambda_1(\tau)$	0.76	0.67	0.65	0.45	0.32
$\lambda_2(\tau)$	0.46	0.41	0.37	0.28	0.19

5. Conclusions

1. Application of termohydrodynamic modeling methods has allowed to generalize the diverse data of full-scale observations and to create the information base containing characteristics $\Xi=\{\xi_i(\vec{r},t)\}$ of the hydrometeorological phenomena on the regular grid $\vec{r}_k$ at the given time moments of t_s.
2. Hydrometeorological fields at fixed $(\vec{r},t)$ designated as $(\bullet)$ are considered as elements of various functional spaces: scalar (atmospheric pressure $p(\bullet)$), Euclidean vector (wind speed $\vec{V}(\bullet)$), function (spectral density of complex waves $S(\omega,\Theta,\bullet)$). In these spaces operations of addition (averaging), multiplication of elements and scalar product have various interpretation.
3. For the description of variability of spatial–time fields canonical variables as coefficients of decomposition in appropriate Hilbert spaces are used. To avoid of multicollinearity basis functions of decomposition are determined by the solution of appropriate Fredholm equations with the help of variational methods.

Acknowledgement

This work is supported by grant INTAS Open 1999 N666.

References

1. Aivazian S.A., Mkhitarian V.S. Applied statistics and essentials of econometrics. Moskow, Book–Publishing Association UNITY, 1998 (in Russian).
2. Anderson T.W. An introduction to multivariate statistical analysis (2nd Ed.) New York, John Wiley, 1984.
3. Balacrishnan, Applied functional analysis. NewYork, John Wiley, 1980
4. Brillinger D.R. Time series. Data analysis and theory. NewYork, Holt, Rinehart and Winston Inc., 1975.
5. Golube G.H., Van Loan C.F. Matrix computations (2nd ed.)., London, John Hopkins University Press, 1989.
6. Gould S.H. Variational methods for eigenvalue problems. University of Toronto Press, 1957.
7. Jonson R.A., Wichern D.W. Applied multivariate statistical analysis (3rd Ed.). Prentice Hall Inc., 1992.
8. Joreskog K.G. Factor analysis by least squares and maximum likelyhood. In: Statistical methods for digital computers. New York, John Wiley, 1975.
9. Hansen I.S. Long–term 3–D modelling of stratification and nutrient cycling in the Baltic sea. Proc. of III BASYS Annual Science Conf., September 20–22, 1999, pp. 31–39.
10. Hotelling H. Relations between two sets of variables. Biometrika, 28, 1936, pp. 321–377.
11. Kahrunen K. Uber lineare Methoden in der Wahrscheinlichkeitsrechnung. Ann. Acad. Sci. Fenn., 37, 1947
12. Kalnay E., M. Kanamitsu, R. Kistler, W. Collins, D. Deaven, L. Gandin, M. Iredell, S. Saha, G. White, J. Woollen, Y. Zhu, A. Leetmaa, R. Reynolds, M. Chelliah, W. Ebisuzaki, W.Higgins, J. Janowiak, K. C. Mo, C. Ropelewski, J. Wang, R. Jenne, D. Joseph. The NCEP/NCAR 40-Year Reanalysis Project. Bulletin of the American Meteorological Society, •3, March, 1996.
13. Komen G.L., Cavaleri L., Donelan M., Hasselmann K., Hasselmann S., Janssen P. Dynamics and modelling of ocean waves. Cambridge University Press., 1994.
14. Loeve M. Fonctions aleatories de second odre. C.R. Acad. Sci. 220, 1945.
15. Lopatoukhin L.J., Rozhkov V.A., Bukhanovsky A.V. The main problems of wind and wave statistics, based on spectral modelling data. Proc. of Coastal Wave Meeting, September, 25–28, Barselona, Spain, 2000, paper 7.5.
16. Rozhkov V.A., Trapeznikov Yu.A. Probabilistic modelling of oceanological processes. S.–Petersburg, Hydromet. P.H., 1990 (in Russian).
17. Ocean wave modeling. Plenum Press. NewYork, 1985
18. Tolman H.L. A third-generation model for wind waves on slowly varying, unsteady and inhomogeneous depths and current //J.Phys.Ocean., 1991, vol.21, N6, pp.782-797.
19. Van Vledder G.Ph., de Ronde J.G. Stive M.J.F. Performance of a stectral wind-wave model in shallow water //Proc. 24th Int. Conf. Coast. Eng. ASCE, 1994, pp.753-762
20. Benoit M., Marcos F., Becq F. Development of third-generation shallow water wave model with unstructured spatial meshing //Proc. 25th Int.Conf.Coast.Eng. ASCE, 1996
21. Lavrenov I.V. Mathematical modeling of wind waves in spatial inhomogeneous ocean. St.Petersburg, P.H. Gidrometeoizdat, 1998
22. Ris R.C. Spectral modeling of wind waves in coastal areas //Communication on Hydraulic and Geotechnical Engineering, June – TUDelft, 1997, N97-4

Numerical Investigation of Quantum Chaos in the Problem of Multichannel Scattering in Three Body System

A.V. Bogdanov, A.S. Gevorkyan, and A.A. Udalov

Institute for High-Performance Computing and Data Bases
P/O Box 71, 194291, St-Petersburg, Russia,
bogdanov@hm.csa.ru, ashot@fn.csa.ru, udalov@fn.csa.ru.

Abstract. The first principle calculations of quantum chaos in the framework of representation constructed by authors for multi-channel quantum scattering was made. Based on intrinsic properties of scattering system the numerical task was divided into independent subtasks and the parallel algorithm for numerical computations was developed and tested on massive-parallel systems *Parsytec CC/16* and *SPP-1600*. This algorithm made it possible to carry out converging computations for three-body problem for any energy. It was shown, that even in the simple case of three-body problem the principle of quantum determinism breaks down in general and one has a micro-irreversible quantum mechanics. The ab initio calculations of the quantum chaos (wave chaos) were carried out on the example of an elementary chemical reaction $Li + (FH) \to (LiFH)^* \to (LiF) + H$.

1 Introduction

At the early stage of quantum mechanics, Albert Einstein wrote the work in which the question was touched, which became a focus of physicists attention several decades later. The question was: what will the classic chaotic system become in terms of quantum mechanics. He has particularly set apart the three-body problem.

In a trial to understand the influence of chaotic dynamical features on quantum quantities calculation the numerical approach plays an important role. In this connection one needs to have high performance algorithms, which allow to make calculations with as small time costs as possible. Numerical results also give sometimes a heuristic ideas, which may be very useful in theoretical considerations.

For the first time the problem of quantum chaos was studied by the authors on the example of quantum multi-channel scattering in collinear three-body system [1,2]. It was shown that this case can be transformed into a problem of a forced unharmonic oscillator with non-trivial time (internal time).

In the present work we discuss the possibilities of chaotic phenomena calculations based on our approach.

V.N. Alexandrov et al. (Eds.): ICCS 2001, LNCS 2073, pp. 473–482, 2001.
© Springer-Verlag Berlin Heidelberg 2001

2 Formulation of the Problem

The quantum multi-channel scattering in the framework of collinear model is considered. As it was shown elsewhere [1,2] the problem of quantum evolution in this case can be strictly formulated as an image point with a mass μ_0 moving over the manifold M, the latter being a stratificated Lagrange surface S_p. At that the motion is related to the local coordinate system moving over S_p. In our case there is standard definition of the surface S_p

$$S_p = \left\{x^1, x^2; P^2\left(x^1, x^2\right) > 0\right\}, \quad P^2\left(x^1, x^2\right) = 2\mu_0\left(E - V\left(x^1, x^2\right)\right), \qquad (1)$$

E and $V\left(x^1, x^2\right)$ being the total energy and interaction potential of the system respectively. The metric on the surface S_p in our case is introduced in the following way: $g_{ik} = P^2\left(x^1, x^2\right)\delta_{ik}$.

The motion of the local coordinate system, it is determined by the projection of the image point moving along the extremal ray $\Im_{ext}$ of the Lagrange manifold S_p. Note, that for scattering problem under consideration there are two extremal rays on the surface S_p : the one corresponding to particle rearrangement and the other corresponding to three free particles. In this paper we study only the first case, namely the one of rearrangement. The quantum evolution of the system on the manifold M is described by the equation (see [2])

$$\left\{\hbar^2 \Delta_{(x^1(s),x^2)} + P^2\left(x^1(s), x^2\right)\right\}\Psi = 0, \qquad (2)$$

with the operator $\Delta_{(x^1(s),x^2)}$ determined in curvilinear coordinates $\left(x^1, x^2\right)$ in Euclidean space R^2, $x^1(s)$ being the x^1-coordinate of image point trajectory. The metric tensor of the manifold M is presented in [2]. Note, that the main difference of (2) from Schrödinger equation comes from the fact, that one independent coordinate, namely, $x^1(s)$ is derived from the set of nonlinear differential equations and is not a natural parameter of the problem. In certain situations it can be a chaotic function.

Our purpose is to find a solution of equation (2) that satisfies the following asymptotic conditions for the total wave function of the system

$$\lim_{(s,x^1)\to-\infty} \Psi^{(+)}\left(x^1(s), x^2\right) = \Psi_{in}\left(n; x^1, x^2\right) + \sum_{m\neq n} R_{mn}\Psi_{in}\left(m; x^1, x^2\right),$$

$$\lim_{(s,x^1)\to+\infty} \Psi^{(+)}\left(x^1(s), x^2\right) = \sum_m S_{mn}\Psi_{out}\left(m; x^1, x^2\right), \qquad (3)$$

where the coefficients R_{mn} and S_{mn} are the excitation and rearrangement amplitudes respectively.

3 Solution of Schrödinger Equation on Manyfold $M\left(\Im(u)\right)$

Taking into account the fact that scattering wave function is located along the reaction coordinate $\Im$ and using the parabolic equation method [3] for such a

problem, we represent the solution of (2) in a form

$$\Psi^{(+)}\left(x^1(s),x^2\right)=\exp\left(i\hbar^{-1}\int\limits_0^{x^1(s)}p(x^1)\sqrt{\gamma_0}dx^1\right)A\left(x^1(s),x^2\right), \tag{4}$$

where $p\left(x^1\right)=P\left(x^1,0\right)$ and γ_0 depends on metric on S_p. After the coordinate transformation in equation (2)

$$\tau=(E)^{-1}\int\limits_0^{x^1(s)}p\left(x^1\right)\sqrt{\gamma_0}dx^1,\quad z=(\hbar E)^{-1/2}p\left(x^1(s)\right)x^2, \tag{5}$$

one gets for the total wave function of three-body system in harmonic approximation [2]

$$\widetilde{\Psi}^{(+)}(n;z,\tau)=\left[\frac{(\Omega_{in}/\pi)^{\frac{1}{2}}}{2^n n!\,|\xi|}\right]^{\frac{1}{2}}\exp\left[i\hbar^{-1}S_{eff}(z,\tau)\right]H_n\left[\frac{\sqrt{\Omega_{in}}}{|\xi|}(z-\eta)\right], \tag{6}$$

where

$$\begin{gathered}S_{eff}(z,\tau)=S_{cl}(\tau)-E_v^i\int\limits_0^{\tau}|\xi|^{-2}d\tau'+\\+\{\dot\eta(z-\eta)+\tfrac{1}{2}\dot\xi\xi^{-1}(z-\eta)^2-\tfrac{1}{2}\dot p p^{-1}z^2\},\\ S_{cl}(\tau)=E\tau-E\int\limits_{-\infty}^{\tau}\{\tfrac{1}{2}[(\dot\eta)^2-\Omega^2(\tau')\eta^2]+F(\tau')\eta\}d\tau',\\ E_v^i=\hbar\Omega_{in}\left(n+\tfrac{1}{2}\right),\quad \Omega_{in(out)}=\lim_{\tau\to\pm\infty}\Omega(\tau).\end{gathered} \tag{7}$$

The functions $\Omega^2(\tau)$ and $F(\tau)$ are defined on S_p and are known, function $H_n(x)$ is a Hermitian polynomial. The function $\xi(\tau)$ is the solution of classical oscillator problem with usual scattering asymptotic condition. As to the function $\eta(\tau)$ it is expressed in terms of $\xi(\tau)$ (see [2]).

4 Transition Amplitude for Rearrangement Processes

One can show that transition probabilities for the reaction $A+(B,C)_n\to(ABC)^*\to(A,B)_m+C$ have the form

$$W_{mn}=|S_{mn}|^2=\frac{(1-\theta)^{1/2}}{m!n!}|H_{mn}(b_1,b_2)|^2\exp\left[-\nu(1-\sqrt{\theta}\cos 2\phi)\right], \tag{8}$$

where the function $H_{mn}(b_1,b_2)$ is a complex Hermitian polynomial, and

$$\begin{gathered}b_1=\sqrt{\nu(1-\theta)}\exp(i\phi),\quad b_2=-\sqrt{\nu}\left[\exp(-i\phi)-\sqrt{\theta}\exp(i\phi)\right],\\ \phi=\tfrac{1}{2}(\delta_1+\delta_2)-\beta.\end{gathered} \tag{9}$$

Denoting $c = (\Omega_{in}/\Omega_{out})^{1/2}$ one has for θ , δ_1, δ_2, β and ν

$$c_1 = e^{i\delta_1} c\,(1-\theta)^{-1/2}\,, \quad c_2 = e^{i\delta_2} c\,[\theta/\,(1-\theta)]^{1/2}$$
$$\theta = |c_2/c_1|^2\,, \quad d = \lim_{\tau\to+\infty} d\,(\tau) = \sqrt{\nu}\exp\,(i\beta)\,. \tag{10}$$

where $d\,(\tau) = (2\Omega_{in})^{-1/2} \int\limits_{-\infty}^{\tau} d\tau^{'}\xi(\tau^{'})F(\tau^{'})$ and the constants c_1 and c_2 enter into an asymptotic expression for $\xi(\tau)$ in the limit $\tau \to \pm\infty$ (see [2]).

5 Numerical Calculations

Schematically algorithm of numerical calculations may be described as a sequence of stages:

- **I** - Lagrange surface construction for the system. The curvilinear coordinate system, within which all further calculations are performed, is introduced on it;
- **II** - classical trajectory problem solution. At this stage the set of four ordinary non-linear differential equations of the first order is solved numerically. Essential initial parameters are collision energy E and oscillator quantum number n. The set is solved by one-step method of 4^{th}-5^{th} order. This method is conditionally stable (by deviation of initial data and rhs) [4], therefore the standard automatic step decreasing method is implied to provide its stability. It's worth mentioning that initial set is degenerate in certain points. To eliminate these the standard σ-procedure with differentiation parameter replacement is performed.
- **III** - the results of classical trajectories calculations are used for calculations of complete quantum wave function in its final state. At this stage, the numerical problem consists in solution of an ordinary non-linear second order differential equation. Numerical investigation of this equation is a difficult task due to non-trivial behavior of differentiation parameter. Differentiation algorithm consists of two stages: 1) construction of differentiation parameter values grid using the results of classical problem calculation and 2) integration of initial differential equation on obtained non-uniform grid by means of multi-step method. Integration stability is provided by selection the integration step in a classical problem, control being performed by means of step-by-step truncation error calculation [4]. The obtained solution of differential equation is approximated in a final asymptotic state in a form of incoming and reflected flat waves superposition;
- **IV** - the results of quantum problem solution are used for obtaining the values of transition probabilities matrix elements and corresponding cross-sections. Calculation of matrix elements for initial oscillator quantum number n and final oscillator quantum number m is performed with the use of expressions presented in [5]. Let's note that transition probability matrix obtained corresponds to one value of collision energy, stipulated at stage **II**;

Let us remind that calculations for steps **II** and **III** are made for specific values of collision energy E and oscillator quantum number of initial state n. Results of these calculations allow to obtain one vector of a reaction cross-section matrix, which corresponds to n. In order to obtain the entire cross-section matrix, calculations at stages **II** and **III** need to be repeated as many times as dictated by the size of reaction cross-section matrix. As a result the entire probability matrix is obtained. The procedure described needs to be repeated for many values of collision energy E in order to enable further integration and velocity constants finding.

Algorithm of numerical calculations allows to perform the parallelization and use the multiprocessor supercomputers with massive parallel architecture for calculations.

Further we will show how the algorithm presented can be parallelized for massive parallel supercomputers with distributed and shared memory.

- *Calculation algorithm for massive parallel systems with distributed memory.* Calculation parallelization procedure is performed by the values of collision energy. Calculation of classical trajectory problem, quantum calculation and transition probability matrix calculation are performed in each of the parallel branches. Let's note that just as in the case on non-parallelized algorithm all calculations from stages **II** and **III** are performed as many times as it is dictated by the size of transition probability matrix. Due to the fact that calculation in each of thee parallel branches represents a separate problem and does not interact with other branches of calculation, the effectiveness of this parallelization algorithm using vs. relatively unparallelized algorithm is nearly proportional to a number of calculation branches, i.e. to the number of computation nodes.
  This algorithm realization was performed on *Parsytec CC/16* supercomputer with massive parallel architecture with distributed memory. As a reaction on which the algorithm was tested, a well studied bimolecular reaction $Li + (FH) \rightarrow (LiFH)^* \rightarrow (LiF) + H$ was taken. The results of testing have shown the calculation efficiency to be nearly proportional to the number of computation nodes.
- *Calculation algorithm for massive parallel systems with shared memory.* Just as in the previous algorithm, the first level of parallelization represents the distribution of calculations among the computation nodes in accordance with the values of collision energy. But, as can be seen from a scheme, in each of the parallel branches there is one more parallelization by the values of oscillator quantum number of the initial state as well. The second parallelization is based upon a fact that for classical trajectory problem calculation the same coefficients, that calculated "on-line", are used for different quantum numbers, thus allowing to make such a parallelization.
  This algorithm was realized on *SPP-1600* supercomputer with massive parallel architecture with shared memory. The results of testing have shown that just as it was expected, the efficiency of calculations is higher than in the previous example.

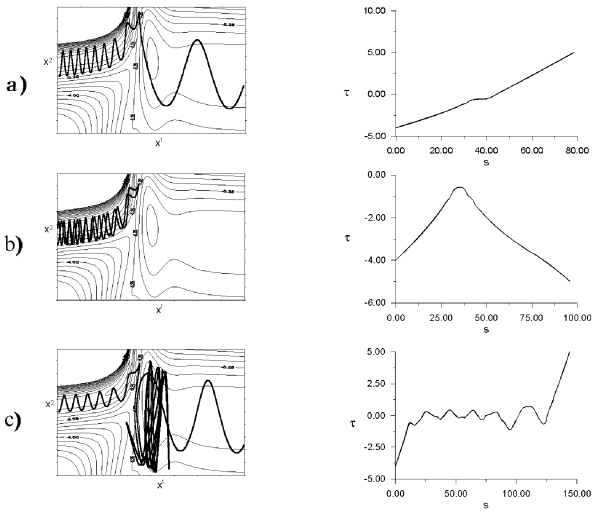


Fig. 1. Geodesic trajectories and internal time dependence on natural parameter s for: a) – direct rearrangement process b) – direct reflection process and c) – rearrangement process going through the resonant state.

Finally we would like to stress one of the important features of parallelization algorithms demonstrated - their scalability. Due to the fact that integration of transition probability matrix and rate constants calculation during stage **V** requires the values of matrix elements for large number of energy values, one can hardly find a supercomputer with an excessive number of computation nodes.

6 Investigation of Classical Trajectory Problem

Numerical calculations were made for the reaction $Li + (FH) \rightarrow (LiFH)^* \rightarrow (LiF) + H$. The potential surface for this reaction was reproduced using the quantum-mechanical calculations carried out in work [6]. Investigation of trajectory problem shows, that, starting from some energy values, internal time $\tau(s)$ from (5), which acts as a chronology parameter in the description of movement along a trajectory, becomes a nontrivial function, having an intricate dependence Fig.1 on the natural parameter (usual time). However, even if $\tau(s)$ is a natural parameter in some region, its derivative $\dot{\tau}(s)$ may have an irregular behavior (see Fig.2). The last property provides sufficient evidence for dynamical chaos existing in the system. As one can see from the Fig.3, the distribution of passed through into the subspace R^2_{out} and reflected back into the subspace R^2_{in} geodesic trajectories with respect to x^1_0 and E for fixed main quantum number n has irregular regions. One gets qualitatively the same picture for other values of n too. Numerical calculations show, that for initial values regions mentioned above

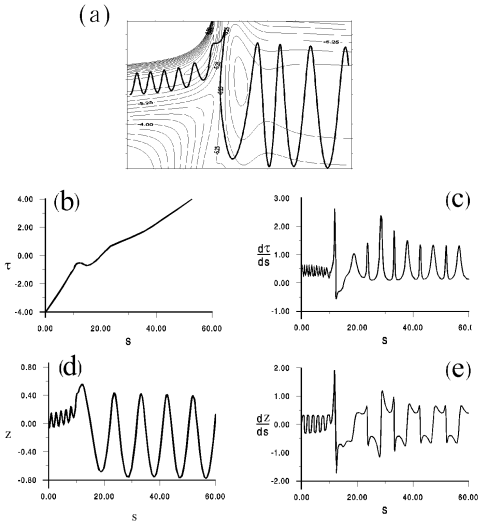


Fig. 2. a) – Geodesic trajectory characterizing direct reaction, b) – corresponding internal time, which is a natural parameter in a wide range, i.e. it is in one-to-one correspondence with parameter s, c) – internal time derivative with respect to s, which has an irregular behavior, d) and e) – show transverse coordinate and its derivative with respect to s respectively.

the main Lyapunov exponent is positive and grows fast, the last fact pointing to exponential divergence of geodesic trajectories. One can see from the results of calculations that the structure of chaotic behavior region is self-similar with respect to scale transformation Fig.3. Chaotic behavior in the classical three body problem disappears when the total energy increases. Note that geometrical analysis of Lagrange manifold for reacting system $Li + FH$ also shows on the possibility of quasicompact submanifold existence, on which intermixing of trajectories may take place.

Similar calculations made for reacting system $N_2 + O$, $N_2 + N$, $N + O_2$, $N + O_2$, $O_2 + O$ show the absence of any irregular regions on the map of passed through and reflected trajectories.

7 Transition Probabilities Calculation for Rearrangement Process

Let us consider the influence of irregular chaotic behavior of classical problem on quantum transition probabilities. It may be illustrated by Fig.4a and Fig.4b which show the dependence of over-barrier transition probabilities in $Li + FH$ system on collision energy E_k^i for fixed phases and quantum numbers. One can

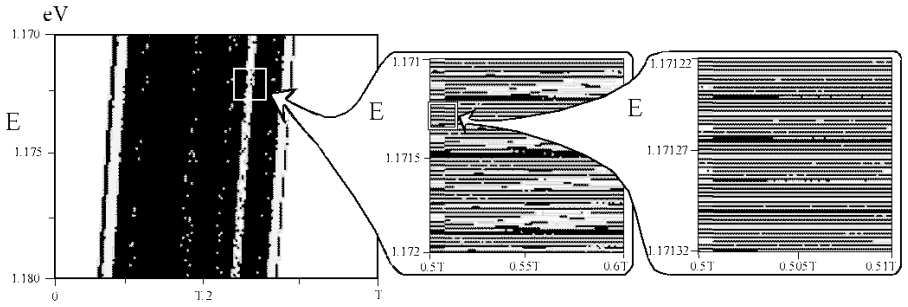


Fig. 3. Irregular map of initial values of the total energy E and initial phase x_0^1 for passed through (white rectangles) and reflected back (black rectangles) geodesic trajectories. T denotes a period of x_0^1.

see, that a small change in initial phase significantly changes the dependencies. In this connection the difficult problem arises to find the measure for the space (map) of passed through and reflected back geodesic trajectories. To calculate

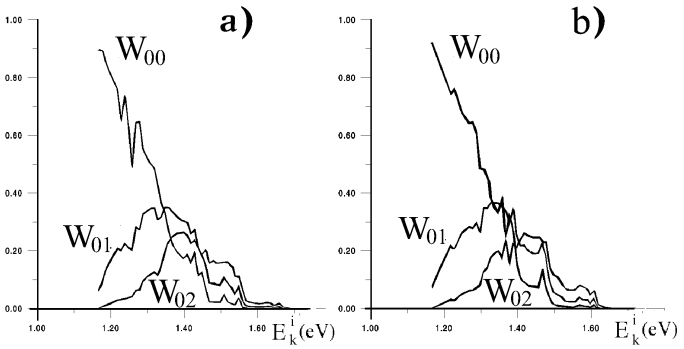


Fig. 4. (a) – Dependencies of transition probabilities W_{00}, W_{01} and W_{02} on collision energy E_k^i for fixed phase x_{0i}^1, (b) – the same dependencies, but calculated for the other (slightly differing from the first one) fixed phase $\bar{x}_{0i}^1 : |\bar{x}_{0i}^1 - x_{0i}^1| = 10^{-5} x_{0i}^1$.

the total mean, giving the final probability for specific quantum transition as depending on energy, one have to average the corresponding quantum probability with respect to (E_k^i, x_0^1) within the range $[\Delta E, \Delta x_0^1]$, where ΔE is a small interval of energies near E_k^i and Δx_0^1 is a period of initial phase. The average process consists in that square $\Delta E \times \Delta x_0^1$ is divided on $n \gg 1$ rectangles, each of them having some phase point x_{0i}^1 inside. Then each rectangle is subdivided by the grid with $M_i = l_i \times k_i$ nodes, l_i and k_i being the number of breaking points for ΔE

and $\Delta x_0^1/n$ intervals respectively. Probability for geodesic trajectory (bearing ray) to pass through the i-th rectangle is calculated by the formula

$$P\left(x_i^1, E_k^i\right) = \lim_{k_i, l_i \to \infty} \left(\frac{N_i}{M_i}\right), \tag{11}$$

where N_i counts how many times the bearing ray passes through into R_{out}^2 subspace. Weighted-mean probability is then calculated as the sum

$$\Delta_{nm}(E) = \lim_{n \to \infty} \left\{ \frac{1}{n} \sum_{i=1}^{n} P\left(x_i^1, E\right) W_{nm}\left(x_i^1, E\right) \right\}. \tag{12}$$

After the averaging with use of (12) the dependence of transition probabilities on collision energy E_k^i for $Li + FH \to LiF + H$ reaction becomes smooth, as one can see from Fig.5.

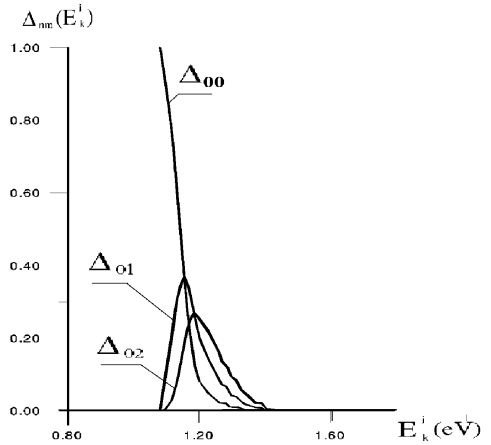


Fig. 5. Transition probabilities dependencies after averaging with respect on phase.

8 Conclusion

The reduction of multi-channel scattering problem solution to trajectory problem allows to use up-to-date multiprocessor computers with massive-parallel architecture and to calculate more trajectories within short period of time. In particular, about 10^5 trajectories were calculated for energy range, within which the resonant state arising is possible. It was shown numerically, that an interval of energies exists in which the internal time dependence on the natural parameter

s may be of oscillatory type. At a small decrease of collision energy, a number of internal time oscillations grows dramatically. In this case the system loses all the information about its initial state completely. Chaos arises in a wave function, which then organizes itself into a new order within the limit $\tau \to \infty$. Mathematically it becomes possible as a result of common wave equation irreversibility by time. One of the numerical results of this work is a chaos map construction Fig.3. There was also numerically shown a strong sensitivity (for chaotic conditions) of quantum transition probabilities dependencies on energy to small changes of initial phase. It was shown that statistical approach must be used to calculate smooth dependencies of probabilities on energy, and the formula (12) for calculation was presented. Let's stress that the result obtained supports the transitional complex theory, developed by Eyring and Polanyi on the basis of heuristic considerations, the essence of the method being statistical description of chemical reactions.

References

[1] A. V. Bogdanov, A. S. Gevorkyan, *Three-body multichannel scattering as a model of irreversible quantum mechanics*, Proceedings of the International Symposium on Nonlinear Theory and its Applications, Hilton Hawaiian Village, 1997, V.2, pp.693-696.

[2] A.V.Bogdanov, A.S.Gevorkyan A.G.Grigoryan, S.A.Matveev, *Internal time peculiarities as a cause of bifurcations arising in classical trajectory problem and quantum chaos creation in three-body system,* Int. Journ. Bifurcation and Chaos, v. 9, N. 12, p. 9-15, 1999.

[3] V.M.Babich, V.S.Buldyrev, *Asimptotic methods in short waves difraction theory*, Nauka, 1972 [Russian].

[4] A. A. Samarskiy, *Introduction to the Numerical Methods* (in russian), "Nauka", Moscow, (1997).

[5] A. N. Baz', Ya. B. Zel'dovich and A. M. Perelomov, *Scattering reactions and Decays in Nonrelativistic Quantum Mechanics* (in russia), "Nauka", Moscow, (1971).

[6] S. Carter, J. N. Murrell *Analytical Potentials for Triatomic Molecules*, Molecular Physics, v. 41, N. 3, pp. 567-581, (1980).

Distributed Simulation of Amorphous Hydrogenated Silicon Films: Numerical Experiments on a Linux Based Computing Environment

Y.E. Gorbachev, M.A. Zatevakhin, V.V. Krzhizhanovskaya, A.A. Ignatiev, V.Kh. Protopopov, N.V. Sokolova, and A.B. Witenberg

Institute for High Performance Computing and Data Bases
Fontanka 118, off. 55, St. Petersburg 198005, Russia
Gorbachev@hm.csa.ru
http://www.csa.ru/comphyga/PECVD

Abstract. Numerical scheme, grid generation and parallelization algorithm are presented for numerical simulation of complex process of silicon-based films growth in plasma enhanced chemical vapor deposition reactors. MPI-based computing environment and advanced interactive software with graphic user interface, real-time visualization system and Web access were recently developed to provide distributed parallel multitask calculation and visualization on Linux clusters. Analysis of system performance and cluster load balance showed the bottlenecks of parallel implementation and the ways of algorithms improvement.

Introduction

Numerical simulation of deposition process of amorphous hydrogenated silicon (α–Si:H) films, the widespread material in modern microelectronics, is an important problem that has been solved over two recent decades. One of the most popular technologies of silicon-based films production is plasma enhanced chemical vapor deposition (PECVD), that has been studied for many years, so the knowledge about physics and chemistry of this phenomenon is rather rich in details. Though dozens of models were developed, describing the system with different levels of abstraction and accuracy, even the most advanced models still can not take into account all the numerous chemical kinetics, plasma physics and transport processes occur in industrial PECVD reactors due to terrific amount of computational time needed for thorough numerical simulation of the problem.

In that way, to find a balance between phenomenon knowledge and computational resources available, not only new physical models are developed, but also new rational numerical algorithms and computational environment are created, adjusted to modern computing facilities.

The project described in this paper is aimed to the development of efficient general purpose computer environment for large parameter space exploration of processes under consideration and to creation of high-performance software for end users working in chemical industry and for scientists studying PECVD processes.

V.N. Alexandrov et al. (Eds.): ICCS 2001, LNCS 2073, pp. 483-491, 2001.
© Springer-Verlag Berlin Heidelberg 2001

Although high performance is the vital factor for simulation of complex-shaped essentially 3D plasma deposition reactors, especially if it should be a short-time forecasting calculations for prediction and controlling of industrial processes, we oriented not to the unique most powerful supercomputers, but to the widely used Linux clusters, accumulated many virtues of parallel computers, and being the cheapest facilities on the factor of FLOPS cost, that is quite important parameter for scientific community. Nowadays, the maturity of clustering software, open sources and easiness of clusters assembling make them more and more popular in high-performance computing.

Model Description

For numerical experiments with cluster computing environment recently developed, 2D approach was chosen for modeling of α−Si:H films deposition from radio-frequency discharge silane plasma. We used the most reliable and well-studied method of computational fluid dynamics for simulation of the physical-chemical processes occur in PECVD reactors, based upon full Navier-Stockes equations. Under the conditions considered (characterized by low Mach and Reynolds numbers), for unsteady laminar flow of viscous compressible chemically active *N*-component gas mixture, these equations in conservation law 2D-form for a Cartesian coordinate system can be written as [1]:

$$\frac{\partial}{\partial t}\int_V U dV + \oint_S (F+G)\cdot dS = \int_V H dV, \tag{1}$$

where t is time and $U = (\rho, \rho u_x, \rho u_y, \rho e, \rho f_1, ..., \rho f_{N-1})^T$ is a vector of conservative variables (here ρ is density, u_x, u_y are velocity components, e is a total energy per mass unit, f_i is mass fraction of i-species), V is arbitrary gas volume and dS is outward vector, normal to the element of surface S which encloses volume V. Flux vectors F and G are associated with convective and viscous transfer correspondingly. Source term H describes species production due to chemical reactions.

For calculation of the electron density, the hydrodynamic model of RF discharge between two planar parallel electrodes was used [2]. The electron and ion continuity equations were solved consistently with the Poisson equation for the electric field distribution.

Boundary conditions taking into account slipping, sticking and deposition effects for velocity, temperature and chemical components concentrations on the wall were applied [3,4].

Numerical Scheme and Grid Generation

Physical processes factorization method [5] was used for solving equations. According to this method the problem was split into inviscid and viscous "steps". At the first one, implicit ENO-scheme [6] was used in combination with bidiagonal

algorithm [7] for implicit increment calculation, and for solving the "viscous" part, a new implicit finite-deference scheme was developed on the basis of [8].

Since an ordinary PECVD reactor has a complex shape (for example, one of the simplest reactors is shown on the figure 7), the multi-block grid generation algorithm was used, when whole computational domain is divided into simple blocks, and then in each block the regular grid is generated. The original method of mechanical analogy with deformed body [9] was applied for generation of non-uniform grids in non-rectangular blocks. The idea of this method is based upon solving equations of elastic body deformation theory:

Let us consider the rectangular plate ABCD made of elastic material with rectangular non-uniform grid plotted on it. Deformation of this plate, leading to the desirable shape of the reactor block, will accordingly deform the grid lines, thus creating the desired curvilinear grid. Figure 1 illustrates this approach. Grid points coordinates in this case are calculated from the elliptic equations of the theory of elasticity [9]. An example of multi-block grid for a complex geometry reactor is shown at the figure 2.

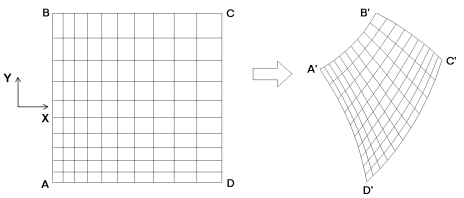


Fig. 1. Plate deformation method for grid generation

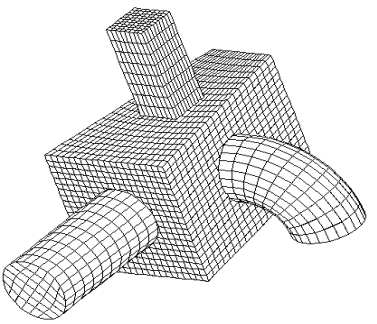


Fig. 2. Example of multi-block grid

Parallelization

Working out the effective computing environment for parallel computing, different levels of parallelization are considered. Because of the large number of parameters which influence upon film deposition rate and properties (pressure, gas and substrate temperatures, flow velocity, reactor volume and configuration, mixture composition and numerous plasma characteristics), for thorough investigation of the system behavior, many tasks with different initial conditions can be run in parallel on different processors, providing thus exploration of a large parameter space. This way of job-level parallelization is the most natural and efficient for many computational fluid dynamic problems which require significant efforts for parallelization of individual program algorithm and code. To utilize the advantages of this approach, a task Manager environment was developed, looking after cluster processors idleness and loading them with new jobs. When one of the processors finished its job, the Manager initiates a process with new initial data set, running over the values of parameters studied.

However, this project is aimed to multipurpose multiscale simulation systems, including 3D simulation and visualization of complex-shaped industrial reactors, that implies terrifically heavy calculations, so parallelization on a task level is also needed. The algorithm and its implementation will be described in the next section.

The last and the most difficult problem of parallelizing and load balancing is the case when all the processors of computing cluster are already loaded with one or few tasks, and a user with the same priority (tasks with the lower priority are simply queue up) starts one or few more program(s). This problem recently arose when the computational system was tested by remote users working with the server cluster through the web-interface. The preliminary results of studying this problem are described in the Results section.

Job-Level Parallelization Algorithm

The natural approach to solve unsteady computational fluid dynamic problem with parallel computers is based on the following two obvious steps:

- Decomposition of computational domain into approximately equal sub-domains so that their number is equal to number of processors.
- At each time iteration each processor calculates only own sub-domain and then sends its results (decrements of variables) to the master processor (with rank 0).

The efficacy of this approach essentially depends on few factors:

- Simplicity of domain geometry, which permits to decompose whole domain into approximately equal sub-domains;
- Spatial uniformity of physical parameters fields and equality of total amount of operations to be done in every sub-domain;
- The ratio of data exchange time between two processes to calculation time for one processor.

Typical domain of PECVD reactor is usually rather complex, so the trivial geometric decomposition makes sub-domains unbalanced in number of grid points and total calculations to be done. A specific decomposition algorithm was developed in order to alleviate this imbalance. It is based upon multi-block regular grid with beam distribution for parallelization.

According to this concept the whole calculation domain is divided into number of primitive blocks which are mere boxes in 2D index space i and j. For example, figure 3 shows a complex domain consisting of nine blocks. Each block is divided into 4-face cells so that in the whole domain has two families of beams: beams in “i” index direction and beams in “j” index direction. Each cell belongs to only one beam of each family and each beam includes a sequence of cells with adjacent faces so that one beam begins and finishes at the physical boundary (fig. 3).

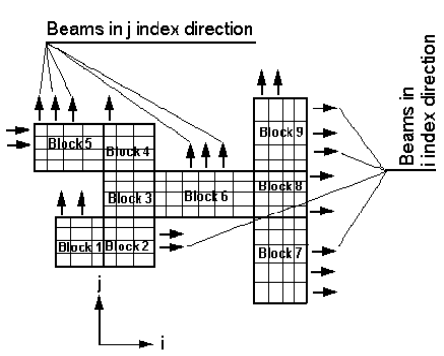


Fig. 3. Nine block domain

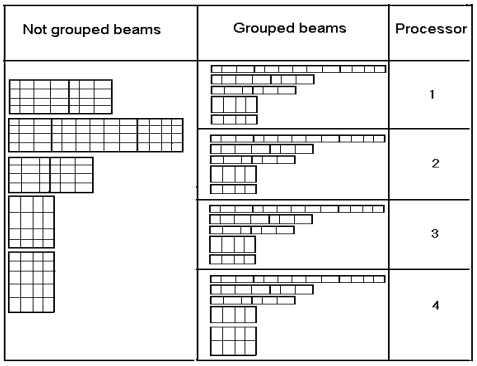


Fig. 4. Beam grouping inside "i" list

This beams decomposition has one important advantage for parallelization. This approach allows using uniform implicit scheme by applying sweep-type algorithms to the single beam and then to distribute beams among the processors.

Thus, the parallel algorithm of calculation at one time iteration boils down to the following steps:

- Formation of two lists of beams in two index directions ("i" list and "j" list) for the given domain.
- Compilation of NP groups of beams inside each of two lists. (NP – number of processors), so that each group has approximately equal number of cells.

  From this point of view the best parallelization efficiency is achieved when the number of cells in every index direction of each block is divisible by *NP*. Otherwise beams are rearranged to meet the condition of approximate equality of number of the cells per every block, but the load balance will not be ideal because of some additional operations on the physical boundaries.

  Figure 4 illustrates group distribution of beams for "i" list of our nine-blocks domain from fig. 3 for four processors. One can see that in this case loading of all processors is approximately equal. Similar grouping is made for "j" list.

- Based on this beam grouping, the work instructions are formed for each processor, listing the beams to be calculated by every processor.
- In the beginning of each time iteration master processor (with rank = 0) sends all necessary information (conservative variables at n-time level) to each slave processor (with rank > 0) according to its instructions. Each slave processor receives the data and computes the beams listed in its instructions.
- After performing its management duties, master processor fulfills its own calculation job and then waits for results (decrements of conservative variables)

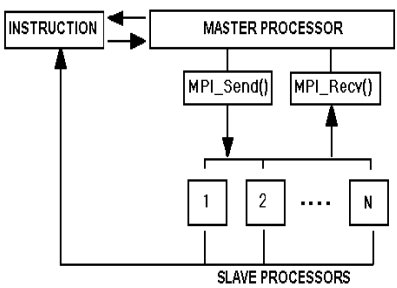


Fig. 5. Parallel algorithm

from the slave processors. On receiving these results the Manager merges all data arrays into one decrements filed and repeats the previous step.

Briefly this algorithm for 1 time iteration is illustrated on the figure 5.

Implementation

For implementation of numerical scheme and parallelization algorithm, MPI message passing interface was used in combination with computational C++ core. To create a high-end software, an advanced graphic user interface was designed. For its realization we used C/C++ programming languages, widespread platform independent GTK+ graphic library and free Glade programming system. The application developed provides users with friendly intuitive interface that allows to control computing process carried out on a server cluster, to visualize numerical results in real-time (or postponed) mode, and to access chemical components data base or results archive. All programs of the software package are platform independent, and were compiled for OS Linux Red Hat 6.2.

To provide access to computational server for users studying PECVD process, the interface of remote user was developed. One of the prevalent technologies used for remote access to distributed computational resources is based upon Web browser as a client application. For this interface, we created HTML pages, Java applets, JavaScripts and CGI scripts. This system, like a local GUI, provides control over calculations and graphical presentation of current physical parameters fields, that is demonstrated on the figure 6.

We have tested the computing environment on two Linux clusters. One consists of four two-processors nodes (each is Intel P-II 450 MHz with 512 Mb RAM) connected with Myrinet network adapter, and the second is 32-processors cluster of Intel P-III 600 MHz with 512 Mb RAM and SCI network communication.

After in-deep analysis of existing API interfaces for MPI library to the Myrinet communication environment, MPI-over-TCP was chosen, which allows to run many programs on cluster nodes simultaneously.

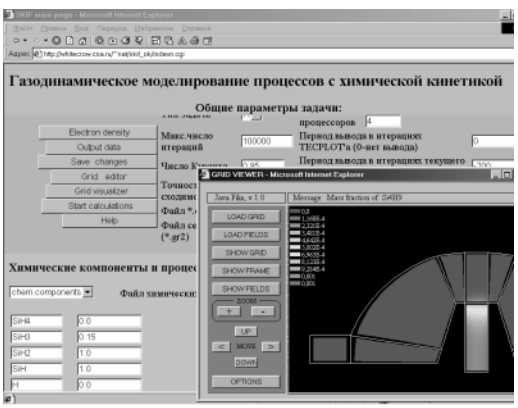


Fig. 6. Remote user interface

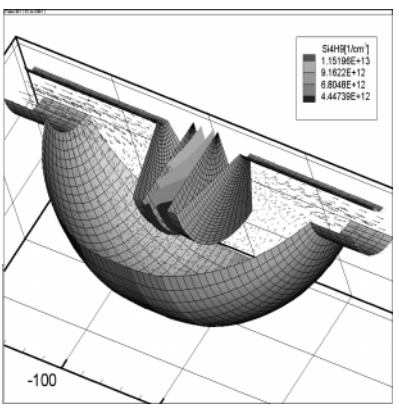


Fig. 7. Simulation results: Si_4H_9 formation

Results and Discussion

A set of numerical experiments was carried out on a Linux-based computing environment, to verify the computational physical-chemical model and to evaluate the efficiency of parallelization algorithm. Reactor geometry, initial gas mixture and plasma discharge parameters were chosen corresponding to the real experimental data available [10]. Further details of the problem formulation can be found in [11].

Influence of different PECVD parameters on film growth and quality was studied, using the job-level parallelization. Running in exclusive mode, this problem did not cause imbalance of cluster processors load, thus a perfect efficiency was achieved with implementation of the task manager controlling processors idleness.

Few distinctive computational results are presented on figures 7 and 8 for the large-scale computational tasks. Here the influence of reactor geometry and pumping path is shown. In the first case (fig. 7, 8a), gas mixture was pumped into the reactor chamber through the inlets located outside discharge area, while in the second case (fig. 8b) mixture flows through a pump tube connected to one of the electrodes. Figure 7 shows the field of Si_4H_9 component (so called higher silane) concentration, and figures 8a,b demonstrate silyl concentration, that is the main component, contributing to the film growth.

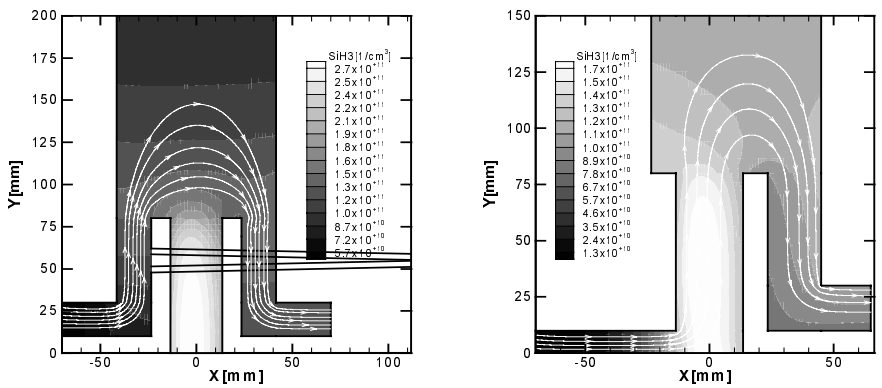


Fig. 8. Simulation results: stream lines and silyl concentration field. a , b - different geometrical configuration of reactors

On the next stage of our work, parallelization efficiency on a task-level was analyzed. Numerous tests showed that due to specificity of the problem, good results of parallel algorithm implementation for an individual task is possible only for a certain range of parameters, when communication overheads are minimal.

Figure 9 illustrates the typical performance result for 'unfavorable' task. Here speedup coefficient was calculated as the ratio of computing time with one processor to computing time for the number processors used. Analysis of communication traffic and workload balance showed, that freezing of speedup with number of processors is caused by too large time of data exchange comparatively with calculation time at every iteration.

One possible way to improve this situation is to replace consecutive exchange between processors with more sophisticated algorithm configured as a processor tree.

Some experiments with different Myrinet communication environment drivers and versions of MPI showed that it is possible to achieve better performance (up to 30 %), but the tendency of freezing speedup coefficient still remains, especially for the middle-scale tasks, where number of beams is not large enough and the fact that their number is not divisible by total number of processors plays a significant role in load imbalance.

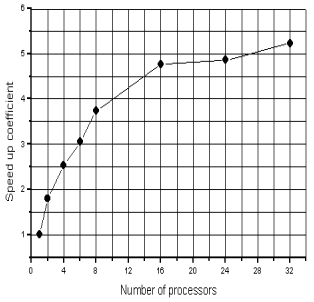


Fig. 9. Speed up coefficient

Besides, it is clear that the faster communications and the larger cash memory would help to overcome the difficulties, but we believe that some advantages also can be won by modification of the algorithms.

Working on the project, we discovered a new problem, when our simulation environment on the cluster was open for remote testing and many parallelized programs were submitted for execution on different number of processors. Serious processors workload imbalance was revealed. Attempts of using standard free-ware utilities, like `queue`, `PBS`, `DQS`, failed because algorithms used there do not suit the specific features of the program. For example, `queue` looks for the least loaded processor and sends a new process there. In the case of a large amount of data to be sent and heavy network traffic, this operation makes the target processor workload even less, and balance manager sends there one more task, collapsing whole calculation process. Under these circumstances, development (or using already produced) dynamic load balancing environment becomes the vital need on the way of creating general purpose efficient software for scientific community.

Conclusion and Future Work

Model and numerical algorithm for simulation of amorphous hydrogenated silicon films deposition were developed, MPI-based computing environment and advanced interactive software with graphic user interface, real-time visualization system and Web access were created, and large-scale parameter space investigations were successfully carried out on two Linux clusters.

Task-level parallelization algorithm using multi-block regular grid with beam distribution was applied for parallel implementation. It provided rather good parallelization efficiency for this class of CFD problem. However, it turned out that due to large communication overheads for most of the tasks considered, system performance almost stops speeding up with increasing of processors number, after a certain 'critical' number. The modification of parallelization algorithm is planned in order to eliminate possible bottlenecks, and some improvements will be made in

initial beams distribution among processors for better load balancing. Possible ways of the communication environment modernization will be also investigated.

Considerable efforts should be made in development of dynamic load balancing environment in order to create a multi-scale multi-user multi-purpose parallel server software to provide an effective utilization of cluster computer power.

References

1. Yu.V. Lapin, M.Kh. Strelets. *Internal flows of gas mixtures*. Moscow, Nauka, 1989 (in Russian)
2. M.I. Zhilyaev, V.A. Schweigert, I.V. Schweigert. *Simulation of RF silane discharge*. Appl. Mech. Tech. Phys. V. 35, N 1, 1994, pp. 13-21
3. M.N. Kogan. *Dynamics of rarefied gas*. Moscow, Nauka, 1967 (in Russian)
4. Yu.E. Gorbachev, M.A. Zatevakhin, I.D. Kaganovich. *Simulation of the growth of hydrogenated amorphous silicon films from an rf plasma.* Tech. Phys. V. 41, N 12, 1996, pp. 1247-1258
5. S.K. Godunov, V.S. Riabenkii. *Difference schemes*. Moscow, Nauka, 1973 (in Russian)
6. Yang J.Y., Hsu C.A. *High-Resolution, Nonoscillatory Schemes for Unsteady Compressible Flows.* AIAA J., V. 30, N 6, 1992, pp. 1570-1575
7. F. Casier, H. Deconinck, Ch. Hirsch. *A class of bidiagonal schemes for solving the Euler Equations.* AIAA J. V. 22, N 11, pp. 1556-1563
8. V.L. Varentsov, A.A. Ignatiev. *Numerical investigations of internal supersonic jet targets formation for storage rings*. Nuclear Instruments and Methods in Physics Research A 413, 1998, pp. 447-456
9. A.A.Ignatiev. *Regular grid generation with mechanical approach*. Mathematical Modelling, V. 12, N 2, 2000, pp. 101-105 (in Russian)
10. G.J. Nienhuis, W.J. Goedheer, E.A.G. Hamers, W.G.J.H.M. van Sark, and J. Bezemer, *A self-consistent fluid model for RF dischargdes in SiH_4-H_2 compared to experiments*. J. Appl. Phys. V.82, 1997, pp.2060-2071
11. Yu.E. Gorbachev, M.A. Zatevakhin, V.V. Krzhizhanovskaya, V.A. Schweigert. *Special Features of the Growth of Hydrogenated Amorphous Silicon in PECVD reactors.* Tech. Phys. V. 45 N 8, 2000, pp.1032-1041

Performance Prediction for Parallel Local Weather Forecast Programs

W. Joppich and H. Mierendorff

GMD – German National Research Center for Information Technology
Institute for Algorithms and Scientific Computing (SCAI)
Schloß Birlinghoven, 53754 Sankt Augustin, Germany

Abstract. Performance modeling for scientific production codes is of interest both for program tuning and for the selection of new machines. An empirical method is used for developing a model to predict the runtime of large parallel weather prediction programs. The different steps of the method are outlined giving examples from two different programs of the DWD (German Weather Service). The first one is the new local model (LM) of DWD. The second one is the old Deutschland Model (DM) which is characterized by more complicated algorithmic steps and parallelization strategies.

1 Introduction

Weather forecast belongs to a class of large applications for parallel computing since more than ten years. The life time of the codes is considerably longer than that of the fast evolving computer systems to be used for weather forecast.

Having in mind a large existing system or an hypothetical one, the key questions which have been posed by the DWD are: Will a one day LM forecast with about $800 \times 800 \times 50$ grid points, using a time step size of 10 seconds, be finished within half an hour. And for the DM: Will a one day forecast with about $811 \times 740 \times 40$ grid points, using a time step size of 7 seconds, be finished within half an hour, too. Additional questions concern the number of processors: for economical and practical reasons the number of processors should not exceed 1024. How should the components of the desired machine look like? Is it necessary to change the code dramatically in order to reach the required speed?

2 Basic Decisions

At first, proper test cases had to be defined. To avoid technical problems with parallel input and output, an artificial topography has been chosen. The initial weather situation is artificial, too. Local models require an update of boundary values from time to time. The new values are generated within the program itself, again in order to avoid either input and output from files or communication with a global model. To have a rather realistic view of the algorithm the decision was to model a 24 hour forecast. By this, especially the time-consuming radiation

V.N. Alexandrov et al. (Eds.): ICCS 2001, LNCS 2073, pp. 492–501, 2001.
© Springer-Verlag Berlin Heidelberg 2001

part is included into the modeling pretty well. For each program test cases are selected in order to derive the model. The final model is applied to predict the run time for the test cases. Where possible, some test cases were not used for modeling but have been left for verification of the performance prediction model. Because the test cases have to run on existing machines and on machines being considerably smaller than the target machine the resolution is far from the finally desired one. The test cases which have been considered both for LM and DM are listed in Table 1.

Table 1. Collection of test cases for LM and DM

	resolution			time-step	processors
	x-dir.	y-dir.	z-dir.	ΔT [s]	
test cases LM	51	51	10, 15, 20	60	1x1 to 4x8
	153	153	10, 15, 20, 25	30	4x4 and 4x8
	325	325	35	30	4x8
target resolution LM	800	800	50	10	
test cases DM	41	37	20,30	15, 45, 90	4x4
	81	73	20	90	2x8, 4x4, 8x2
					16x4, 4x16, 8x8
	81	73	30, 40	90	4x4
	121	109	30	90	4x4, 8x8
	271	244	40	90	8x8
target resolution DM	811	741	40	7	

3 Analysis of the Parallel Programs

Partial differential equations are the mathematical background of weather prediction models. The equations are discretized in space and in time. The programs are written in FORTRAN 90 and the message passing interface MPI is used for communication on parallel machines. The basic concept of parallelization is grid partitioning. This concept is realized in a straight forward manner, including the usual overlap region for each partition both in the LM and in the DM. Nevertheless, the mapping of the different data fields to logical processes is slightly different within the two programs. In Figure 1 the 2D-partitioning strategy and the mapping to processes is shown both for the LM and for the DM. The number of inner points per partition and the logical process numbers are given. For the DM (right picture) the process identification number p_{id} is determined by the given pair of indices (p_{idx}, p_{idy}): $p_{id} = nproc_x(p_{idy} - 1) + p_{idx}$ where the number of participating processes $nproc$ is the product of $nproc_x$ and $nproc_y$. The interior overlap areas are not shown. But this overlap area in practice enlarges for instance a DM partition by two lines and columns in each direction. This type of partitioning is used everywhere within the LM and in most parts of the DM.

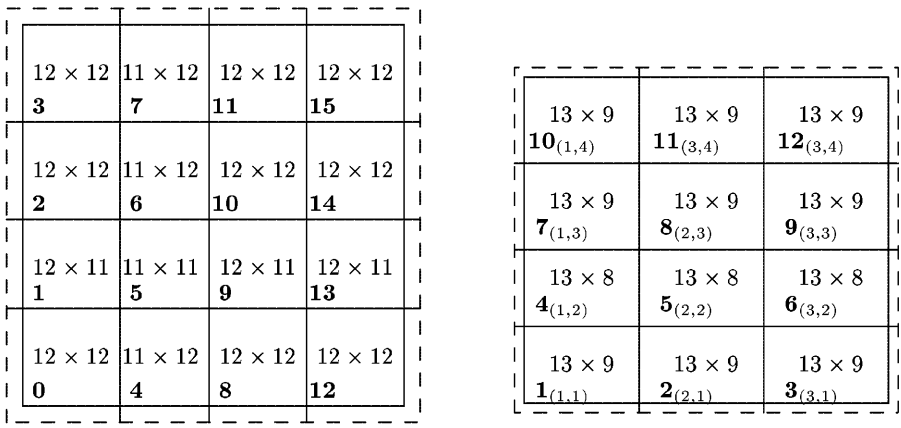


Fig. 1. LM grid (51×51, left) partitioned for 4×4 processes, DM grid (41×37, right) partitioned for $12 = 3 \times 4$ processes

The general approach to solve the discrete equations derived from the continuous model consists of an iterative sequence of several phases like dynamics computations, physics computations, FFT, Gauss-elimination, local communication, global communication, input/output of results, and transposition of data within the system of parallel processes. The LM equations are discretized explicitly in time and require no special solver for large systems of algebraic equations. The DM instead uses a semi-implicit method which leeds to a discrete equation of Helmholtz type. The corresponding discrete equation is solved by a well-established algorithm using Fast Fourier Transformation (FFT) in combination with Gauss elimination (GE). The data-partitioning for these cases is described in Figure 2. It is necessary to switch between the different partitioning strategies within the algorithm (transposition). Such a transposition is a very challenging task for the communication network of massively parallel computers.

The programs have been instrumented such that they provide with detailed timing information about every essential part of the program. From this information the computational complexity of the two main parts of each program (dynamics and physics) have been modeled. The model depends on critical parameters of the parallel application like size of the partition, relative position of a partition within the partitioning structure (boundary process, interior process), and time steps. The analysis of the parallel programs has led to a complete set of information about the communication pattern, the communication complexity, and the communication frequency of the programs. As an example, the result of the analysis is given for the subroutine TRANSPOSE of the program DM. This subroutine, which is called four times per time step, exchanges the data between the different processes when switching from 2D-partitioning to FFT row partitioning, to GE column partitioning, and back to 2D-partitioning. FFT and GE belong to the dynamics part of the DM. In Table 2 myi, myj, and myk

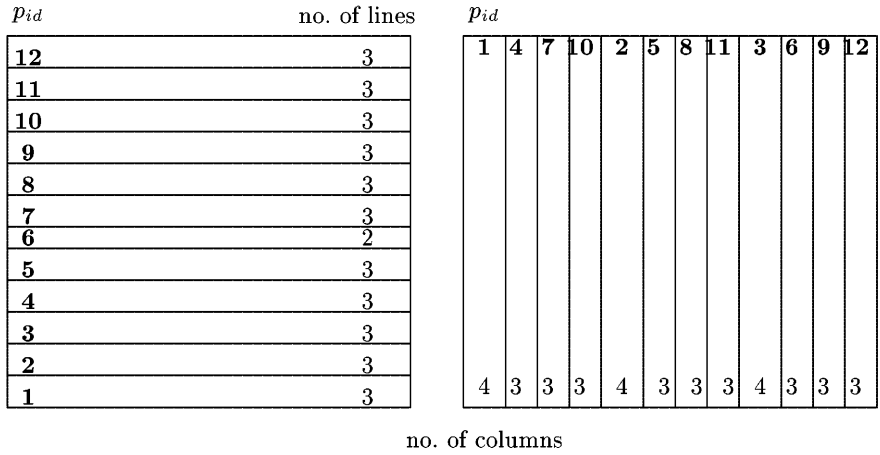


Fig. 2. DM grid (41×37) distributed to 3×4 processes for the steps FFT (left) and GE (right)

denote the local size of the 2D partition in x, y, and z direction, respectively. $rfft$ and cge are the rows and columns of a process for FFT and GE (the index pp denotes the same quantity for the partner process).

Table 2. Messages which are sent by TRANSPOSE in DM

no. of calls per day: $\frac{24 \cdot 3600}{\Delta T}$ action	total no. of processes	messages per process	message length
2D $\rightarrow$ 1D-FFT	$nproc_x \times nproc_y$	$nproc_x - 1$	$myk \cdot rfft_{pp} \cdot myi$
1D-FFT $\rightarrow$ 1D-GE	$nproc_x \times nproc_y$	$nproc - 1$	$myk \cdot rfft \cdot cge_{pp}$
1D-GE $\rightarrow$ 1D-FFT	$nproc_x \times nproc_y$	$nproc - 1$	$myk \cdot rfft_{pp} \cdot cge$
1D-FFT $\rightarrow$ 2D	$nproc_x \times nproc_y$	$nproc_x - 1$	$myk \cdot rfft \cdot myi_{pp}$

The different phases are synchronized because of data dependences and by local and global communication phases. The LM also contains MPI barrier calls for synchronization and the DM uses wait-all at the end of subroutines which realize the exchange of data with neighboring processes. Therefore, an additive model can be used to estimate the overall runtime by adding the runtime of all the single phases: dynamics, physics, FFT (only DM), GE (only DM), and communication which includes the time for buffering of data into a linearly ordered send buffer (and similar for the receive).

4 Modeling Computational Time

For estimating the runtime of the dynamics part of the LM it turned out that the numerical effort is mainly depending on the number of interior grid points. The upper left picture of Figure 3 shows this for the first hour of a low resolution experiment. There is some effort related to exterior boundary points which can be identified by analyzing several examples numerically. But this has almost no influence. The dynamics computing time depends linearly on the number of levels in vertical direction and linearly on the time step size as the lower pictures of Figure 3 show for the DM.

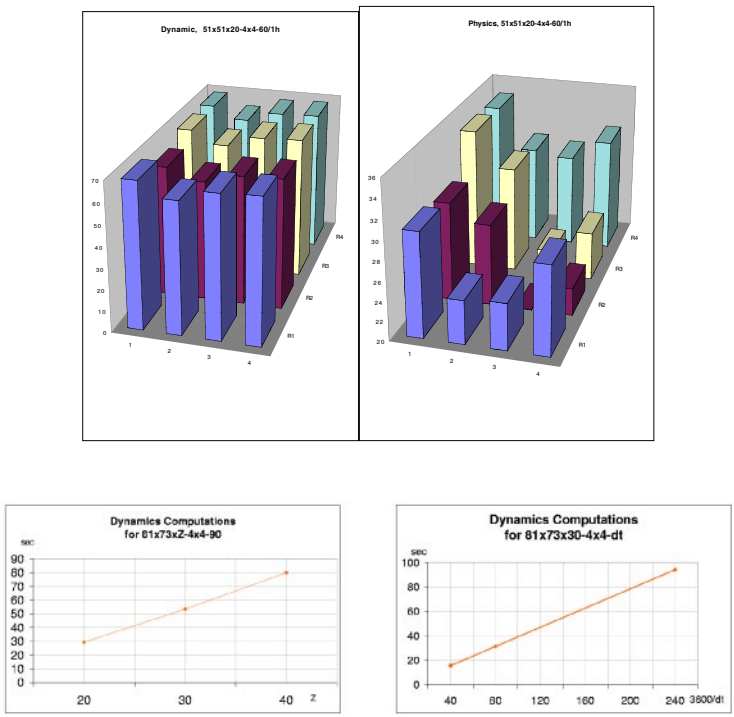


Fig. 3. Distribution of computational time for the first hour. LM dynamics (upper left) and LM physics (upper right) on 4×4 processors, low resolution. Linear dependence of the dynamics computation time for DM (levels, lower part left and time-step size, lower part right).

The computational time is approximated by a multi-linear function in the number of interior grid points (and derived quantities) and the number of time steps. The coefficients of the multi-linear function have been determined by a least squares method in order to match all considered measurements. Because it

is intended to extrapolate the performance of very large application cases running on very large computing systems on the basis of measurements of relatively small examples running on small parallel systems the leading term of the function has to be determined as exact as possible.

The runtime of physics computations depends on the number of grid points, the hour, and on the local weather situation. Especially the last mentioned effect causes non-balanced load, see the upper right picture in Figure 3. To avoid the dependence on the hour it is possible to accumulate all values of a day. Since the different numerical processes of the parallel codes are synchronized by data dependences and calls of some synchronizing MPI-constructs, the additive model has been justified and therefore the maximum runtime among all processes is of main interest. However, this maximum should not be caused by the number of grid points but by the local weather. The physics measurements are related according to the effective number of grid points before using them for estimating the runtime of physics computations. In principle, it is the ratio of physics and dynamics computation time which is approximated in order to get the computational time for the physics part of the programs [4].

5 The Machine Models

The machine models are based on measurements on existing machines. The benchmarking itself took much time and new questions arose while evaluating the results. The large amount of data required a careful analysis and not all of the data was reliable at all. The programs behaved differently on different architectures. One example is the influence of cache effects. This was observed both for LM and DM. Therefore, if necessary, the model includes for each part of the algorithm a cache factor which represents the effect of slow down by L2-cache misses. This factor depends on the local partition size, on the size of the cache and on the algorithm. It is desirable after all that the local partition fits into the cache. Figure 4 shows these slow-down factors both for the dynamics computation and the Gauss elimination in DM. This heuristic approach is strongly depending on the program under consideration and is reliable only in the range of the measured points. In addition to this cache effect the memory access is also included into the models. The time for copying data from 3D-data structures into linearly organized send buffers and back from receive buffers into 3D-arrays is not necessarily neglectable. The model for buffering is based on measurements with a program which uses a similar data structure for data access and storing as in the applications themselves. The analysis of the communication pattern combined with measurements was used to set up a formula to compute the time for local communication. Models for the barrier- and reduce-function, respectively, have been developed from measurements, too. They also reflect the underlying implementation of these MPI-constructs. The knowledge about the frequency and the required action of these events finally allows to estimate the time for these parts of the programs (see Tables 6 and 7).

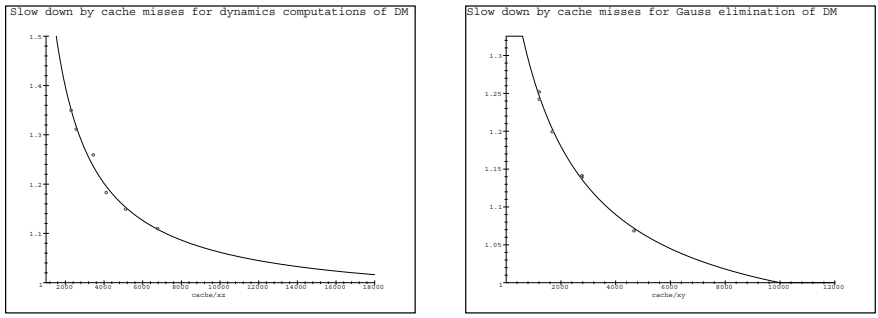


Fig. 4. Model for slow down of the DM by cache effects; dynamics left, GE right, Origin2000

6 Verification of the Models

The initial model for the LM has been developed on a SP-2 with 32 processors using 9 out of 13 parallel runs. Due to the needs of vector machines the models have been ported to an SX-4 using 3 calibration runs to adapt the model parameter. The vector start-up and the computational effort per grid point had to be included. This could be done because the main direction of vectorization was known. Further, the partitioning has to be adapted to the direction of vectorization in order to get long vectors. But none of the application programs is particularly tuned for vector processors, yet.

For proving the reliability of the models, the deviation of model evaluation and runtime values for a collection of test cases is shown. Some of the test cases are used for model development, others are exclusively used for testing the performance prediction. The values are given in Table 3 for the LM on an SP-2. Similar results are given for the LM running on an SX-4 in Table 4. All times concern 24 hour simulations. The different cases are characterized by the resolution in x, y, and z direction, the number of processors, $nproc_x \times nproc_y$, and the time step size in seconds ($x \times y \times z - r \times s - \Delta T$).

Table 3. Comparison of measured and estimated timing values for the LM on SP-2

case	runtime [h]		deviation [%]
	measured	estimated	
$51 \times 51 \times 20 - 1 \times 1 - 60$	8.75	8.85	1.17
$51 \times 51 \times 20 - 2 \times 2 - 60$	2.42	2.48	2.41
$153 \times 153 \times 20 - 4 \times 4 - 30$	11.20	11.30	0.85
$153 \times 153 \times 25 - 4 \times 8 - 30$	7.42	7.47	0.78
$325 \times 325 \times 35 - 4 \times 8 - 30$	43.50	45.87	5.44

Table 4. Comparison of measured and estimated timing values for the LM on SX-4

case	runtime [h]		deviation [%]
	measured	estimated	
$153 \times 153 \times 20 - 1 \times 1 - 30$	1.23	1.32	7.40
$153 \times 153 \times 20 - 1 \times 4 - 30$	0.35	0.34	-0.99
$255 \times 255 \times 25 - 1 \times 1 - 30$	3.56	3.77	5.86
$255 \times 255 \times 25 - 1 \times 4 - 30$	1.02	0.96	-6.07
$255 \times 255 \times 25 - 1 \times 12 - 30$	0.37	0.33	-11.24

Models of the DM have originally been developed for the VPP700 as well as for an Origin2000. The verification runs for the DM on a VPP700 show a maximum deviation from measurement in the range of less than ten percent. For a twelve hour forecast (the above tables show results for 24 hour simulations) on an Origin2000 Table 5 shows an acceptable agreement between estimated and measured time.

Table 5. Comparison of measured and estimated timing values for the DM on Origin2000

case	runtime [h]		deviation [%]
	estimated	measured [s]	
	(12 hrs)	(12 hrs)	
$41 \times 37 \times 20 - 1 \times 1 - 90$	216,54	222,62	-2,73
$41 \times 37 \times 30 - 4 \times 4 - 90$	33,73	32,93	2,44
$81 \times 73 \times 20 - 8 \times 8 - 90$	31,19	32,25	-3,29
$81 \times 73 \times 20 - 16 \times 4 - 90$	34,72	37,41	-7,18
$121 \times 109 \times 30 - 8 \times 8 - 90$	77,84	80,84	-3,72
$271 \times 244 \times 40 - 8 \times 8 - 90$	468,78	447,41	4,78

7 Application of the Models

As already mentioned, the desired resolution for the LM is 800×800×50 with $\Delta T = 10$ seconds. The extrapolation to this final resolution both in space and time and considering the defined set of parameters for the test case neglects the fact that not each of the systems having in mind allows an extremely large configuration. Therefore the 1024-processor system is assumed to be a cluster architecture, if necessary. The parameters for the cluster network communication can be specified.

Applying the model to a T3E-600 the prediction for the LM delivers an estimated runtime of more than 5 hours. The expected distribution of work is specified in Table 6 both for 1024 and 2048 processors.

Table 6. Runtime estimation for the final resolution on T3E-600

case	runtime	distribution of work [%]						eff.
	[h]	dynam.	physic	comm.	buffer.	barrier	reduce	
800×800×50−32×32−10	5.62	54.36	41.21	0.75	3.61	0.06	0.01	0.82
800×800×50−32×64−10	3.08	53.29	41.06	1.11	4.40	0.11	0.03	0.75

This shows that the processor speed compared to the T3E-600 has to be increased by a factor of about 11 in order to reach the requirements of the DWD for the LM. The results also show that the T3E interconnect network is powerful enough to work with processors having the required speed. We have applied the model to a currently available Origin2000 (195 MHz processors, Table 7). Because of the shared memory architecture we had to assume this machine to consist of several (4×4 and 8×8) Origin2000 systems with 64 shared memory processors each.

Table 7. Runtime estimation for the final resolution on an Origin2000 cluster architecture

case	runtime	distribution of work [%]						eff.
	[h]	dynam.	physic	comm.	buffer.	barrier	reduce	
800x800x50-32x32-10	5.53	76.34	17.73	4.19	1.48	0.25	0.01	0.83
800x800x50-64x64-10	1.78	71.03	16.76	8.67	2.55	0.94	0.05	0.64

8 Conclusion

Two performance prediction models (for LM and DM) have been developed both for vector machines and for clusters of shared memory architectures. Although the old DM is no longer used for operational purposes it may serve as benchmark because of its sophisticated algorithm and due to the changing partitioning strategy within different algorithmic steps. The original version of the DM contains the data transposition as described in [1] - [3]. This type of transposition turns out to be the bottle-neck on large systems because the number of messages then increases with the square of the processors used (a transposition strategy which linearly depends on the number of processors is assumed for the model). Further improvement should be made by parallel FFT and Gauss-elimination. Otherwise the degree of parallelism would be limited by the 1D-partitioning of these computational steps.

The initial version of the LM performance prediction model (1998) predicted that no 1024-processor system of at that time existing processors would reach the goal of finishing a one day forecast using the required resolution within half

an hour computing time. But it was observed that a Cray T3E-like network is powerful enough to work with processors which are up to ten times faster. Such a T3E-like system with nodes of approximately seven to ten GigaFlop has been estimated to be able to satisfy the required condition.

After adaptation of our model to the needs of vector machines (1999) and choosing an appropriate partitioning (long vectors) large systems of about 512 vector processors are expected to be close to the requirements. It is still an open question which performance parameters the architecture at the DWD will show at the end of 2001.

The development of the performance prediction model has led to improvements of the codes themselves. The application of the models allows to estimate the influence of hardware parameters of future computer architectures to operational weather forecast.

Acknowledgments

This work has been initiated by G.-R. Hoffmann from DWD. E. Krenzien and U. Schättler advised us how to use the codes. We had helpful discussions with E. Tschirschnitz (NEC) and R. Vogelsang (SGI). K. Cassirer (GMD), R. Hess (GMD, now DWD), and H. Schwamborn (GMD) substantially contributed to this work.

References

1. D. Dent, G. Robinson, S. Barros, L. Isaksen: *The IFS model – overview and parallel strategies.* Proceedings of the Sixth Workshop on Use of Parallel Processors in Meteorology, ECMWF, 21–25 November 1994.
2. Foster, I., Gropp, W., Stevens, R.; *The parallel scalability of the spectral transform method.* Monthly Weather Review 120 (1992) 835 – 850.
3. U. Gärtel, W. Joppich, A. Schüller, *Portable parallelization of the ECMWF's weather forecast program*, Arbeitspapiere der GMD 820, GMD, St. Augustin, 1994.
4. H. Mierendorff, W. Joppich, *Empirical performance modeling for parallel weather prediction codes*, Parallel Computing, 25 (1999), pp. 2135-2148.
5. W. Gropp and E. Lusk, *Reproducible Measurements of MPI Performance Characteristics*, Argonne NL, `http://www.mcs.anl.gov/mpi/mpich/perftest` or by ftp from `ftp://ftp.mcs.anl.gov/pub/mpi/misc/perftest.tar.gz`.
6. P. J. Mucci, K. London, J. Thurman, *The MPBench Report*, November 1998, `http://www.cs.utk.edu/~mucci` or by ftp from `ftp://cs.utk.edu/thurman/pub/llcbench.tar.gz`.
7. *Pallas MPI Benchmarks - PMB, Part MPI-1*, Revision 2.1, 1998, `http://www.pallas.de/pages/pmbd.htm`.
8. R. Reussner, *User Manual of SKaMPI (Special Karlsruher MPI-Benchmark)*, University of Karlsruhe, Department of Informatics, 1999.

The NORMA Language Application to Solution of Strong Nonequilibrium Transfer Processes Problem with Condensation of Mixtures on the Multiprocessor System

A.N. Andrianov[1], K.N. Efimkin[1], V.Yu. Levashov[2], and I.N. Shishkova[3]

[1] The Keldysh Institute of Applied Mathematics, Russia,
efi@a5.kiam.ru
[2] Institute for High Performance Computing and Data Bases, Russia
levashov@mcsa.ac.ru
[3] Moscow Power Engineering Institute (Technical University), Russia
irina@mcsa.ac.ru

Abstract. The application of NORMA language for problem of strong nonequilibrium transfer processes is discussed. The parallel algorithm for problem solution is created. This algorithm was realized on the multiprocessors system.The result of calculation and efficiency of program are presented.

1 Some Properties and Facilities of the NORMA Language

The declarative NORMA language [1]- [3] formalised the mathematical specification resulting from discretization of continuous differential equations. Thus this is the language of extremely high level and a specification of a computational task is turned into an executable program automatically by the translator-synthesiser for the NORMA language. This language was created to specify generic grid-based solutions to problems in applied mathematics but the area of its application turned out to be wider. Now the NORMA language is at the stage of practical use.

Note that the specification of a task solution in NORMA mentions only those rules (constraints) which must be met by the values of the variables and besides specification has no embedded memory representations and few of usual elements of programs (e.g., no control statements). Only the Norma translator needs to know about memory, processors, caches and the other hardware paraphernalia that make most programs in general purpose programming languages so hard to port to new computing environments.

It is important to note that there are no extra dependencies in the NORMA specification though they are usually imposed in programming especially at the stage of algorithm optimisation. These links often limit the possibilities of parallelising. For instance the construction COMMON in the Fortran language or indirect addressing usually limits automatic parallelization of the programs. From

V.N. Alexandrov et al. (Eds.): ICCS 2001, LNCS 2073, pp. 502–510, 2001.
© Springer-Verlag Berlin Heidelberg 2001

this point of view the NORMA language has one more advantage: it is the language with single assignment. This fact is known to be very important for automatic parallelising.

The scheme for parallelising a solution used during a NORMA translation is briefly as follows. A graph of data dependencies is built after their analysis. A level-parallel graph of the algorithm is constructed to satisfy all dependencies with the maximum possible (ideal) parallelism. That graph is projected onto the architectural model of the target computer system. In constructing a projection, the memory model (distributed memory or shared memory), the number of the processors, and various component bandwidth factors are taken into consideration. Different optimizations are performed, of various kinds to attempt to solve performance problems related to the granularity of parallelism, load balancing on all available processors etc.

From our point of view, instead of changing a sequential version of a program to a parallel one or directly designing a parallel program, there is a more promising third way when the original statement of a problem may be realized as both parallel and sequential variants. This third approach is based on a key principle: *when formulating a new task do not impose extra constraints - later you may face an environment where they cause inconvenience and inefficiency.* Fortunately, mathematical specifications almost always adhere to this principle.

The NORMA language required new algorithms for translation and parallelising. The major results are given in [4]. Norma language compiler can generate target parallel programs in Fortran with PVM or MPI libraries or OpenMP Fortran or serial Fortran programs.

Some key features of the NORMA language are named below.

NORMA contains features for both *common* mathematical notions (e.g. integer, real numbers, vectors, matrices, functions etc.) and the notions *typical* for the given application domain (e.g. grid, index space, grid function, iteration on time and space).

In NORMA the notion *domain* represents an index space. It contains integer sets $i_1, \ldots, i_n, n > 0$ each of them is a co-ordinate of a point in n- dimensional index space. A unique index name is given to each coordinate axis of n-dimensional space. Domain may be *conditional* and *unconditional*. Conditional domain consists of the points from index space which number and coordinates may be changed depending on meeting the *conditions on domain*. An unconditional domain consists of a fixed number of points in an index space at coordinates that can be determined during translation.

A *one-dimensional domain* sets a range of points along some coordinate axis of index space, for example: **RegionK: (k=1..15)**. A *multidimensional domain* is a domain product built by operation "**;**". For example, two-dimensional domain **Oij** is a product of domains **Oi** and **Oj**: **Oij: (Oi;Oj)**, where domains **Oi** and **Oj** can be declared as **Oi: (i=1..N)** and **Oj: (j=2..M)**. Possible *modification* to a domain includes adding or deleting some points and changing the range. Domain may be *conditional* and *unconditional*. Conditional domain consists of the points from index space which number and coordinates may be changed

depending on meeting the *conditions on domain*. An unconditional domain consists of a fixed number of points in an index space at coordinates that can be determined during translation.

In NORMA *scalar variables* are simple variables but *variables defined on domain* are vectors, arrays and matrices. Declaration of the variable sets its type - **REAL**, **INTEGER**, or **DOUBLE** - and, if it is a variable on a domain, indicates the domain of points where variable values may be computed. For example, declaration **VARIABLE First**, **Last DEFINED ON Oij** defines variables **First**, **Last** on domain **Oij**; it means that the values may be assigned to these variables in every point of domain **Oij** for $i = 1, \ldots, N, j = 2, \ldots, M$.

Calculating formulae obtained by technical expert are usually written in the form of *relations*. For example, calculating formulae for the solution system of linear equations $Ax = B$ has the form:

$$
\begin{array}{ll}
m_{0,i,j} = a_{i,j}, & j = 1, \ldots, N, i = 1, \ldots, N; \\
r_{0,i} = b_i, i = 1, \ldots, N; & \\
m_{t,t,j} = m_{t-1,t,j}/m_{t-1,t,t}, & j = 1, \ldots, N, i = 1, \ldots, N; \\
r_{t,t} = r_{t-1,t}/m_{t-1,t,t}, & i = 1, \ldots, N; \\
m_{t,i,j} = m_{t-1,i,j} - m_{t-1,i,t} * m_{t,t,j}, & j = 1, \ldots, N, \\
 & i = 1, \ldots, t-1, t+1, \ldots, N; t = 1, \ldots, N; \\
r_{t,i} = r_{t-1,i} - m_{t-1,i,t} * r_{t,t}, & i = 1, \ldots, t-1, t+1, \ldots, N; t = 1, \ldots, N; \\
x_i = r_{N,i}, & i = 1, \ldots, N;
\end{array}
$$

Extract from the NORMA program is given below:

Example of a NORMA Program

```
Ot:(t=0..n) . Oi:(i=1..n) . Oj:(j=1..n).
Oij:(Oi;Oj) . Otij:(Ot;Oij) .
Oti:(Ot;Oi) . Otij1:Otij/t=1..n.  Oti1:Oti/t=1..n.
        DOMAIN PARAMETERS n=10.
VARIABLE a DEFINED ON Oij.   VARIABLE m DEFINED ON Otij.
VARIABLE b DEFINED ON Oi.   VARIABLE r DEFINED ON Oti.
        INPUT a ON Oij,  b ON Oi.
        OUTPUT x ON Oi.
FOR Otij/t=0 ASSUME m = a.
FOR Oti/t=0 ASSUME r=b.
Otij11, Otij12:Otij1/i=t.   Oti11, Oti12:Oti1/i=t.
FOR Otij11 ASSUME m = m[t-1,i=t]/m[t-1,i=t,j=t].
FOR Oti11 ASSUME r = r[t-1,i=t]/m[t-1,i=t,j=t].
FOR  Otij12 ASSUME m = m[t-1]-m[t-1,j=t]?m[i=t].
FOR Oti12 ASSUME r = r[t-1]-m[t-1,j=t]?r[i=t].
FOR Oi ASSUME x = r[t=n].
```

Necessary computations are specified by **ASSUME** operator:
FOR *domain* **ASSUME** *relation*.

This operator is a key-feature of the NORMA language. The relation gives the rule for computing the variable's value from the left part by the values of

the variable from the right part. It also gives the index dependencies between the variables. Values for the left variable must be computed at all points of the domain. The rule for each computation is defined very precisely but the computations do not need to occur where given. The program does not specify the mode (parallel or serial) or order for computations. It just tells what value relations must be preserved. Specific computations must be done only soon enough to determine values that depend upon them.

Indices with no offsets in the formulae notations may be omitted because they are automatically restored by the translator in analysis of the program.

The conditional domain definition, **Otij11**, **Otij12 : Otij1/i=t** determines two disjoint sub-domains **Otij11** and **Otij12**. The first consists of points from domain **Otij1** where condition **i=t** is true; the second, points where $\mathbf{i} \neq \mathbf{t}$. In general, a condition is a logical expression. The differences in two given above ways of calculating formulae representation are only in the form (index representation, linearity of the specification) but they are equivalent in their contents.

2 Description of the Application

Norma language and Norma system was used to create parallel program implementation for the problem of strong nonequilibrium transfer processes with condensation of different mixtures on the surfaces.

Continuum media methods give a good description of different phenomena in conditions which are characterized by small deviation from the state of thermodynamic equilibrium. By nonequilibrium growth the study of transfer processes should be made by the help of molecular kinetic theory basing on the Boltzmann equation.

Boltzmann kinetic equation (BKE) for two-dimensional non-steady statement for two- component mixture has form:

$$\begin{aligned} \frac{\partial f_A}{\partial t} + \xi_x \frac{\partial f_A}{\partial x} + \xi_y \frac{\partial f_A}{\partial y} &= J_{AA} + J_{AB} \\ \frac{\partial f_B}{\partial t} + \xi_x \frac{\partial f_B}{\partial x} + \xi_y \frac{\partial f_B}{\partial y} &= J_{BB} + J_{BA}, \end{aligned} \tag{1}$$

where $f = f(x, t, \xi)$ is velocity distribution function, $\xi(\xi_x, \xi_y, \xi_z)$ - molecular velocity, t - time, x, y - Cartesian co-ordinates, J_{AA} - collision integral describing interactions between molecules of component A, J_{BB} - collision integral describing interactions between molecules of component B, J_{AB} and J_{BA} - between A and B molecules.

In writing the expression for each collision integrals we have used the notations introduced by [5].

$$J = \iint\limits_{\Omega} \iiint (f' f_1' - f f_1) \, |\xi - \xi_1| b d\Omega, \quad d\Omega = db \, d\epsilon \, d\xi_1. \tag{2}$$

The method of direct numerical solving of Boltzmann equation [5] is used which, the authors believe, is one of the most correct methods in kinetic solving of the processes characterized by strong nonequilibrium. The Boltzmann kinetic equation, from the physical viespoint, adequately describes gas flows with high deviation from local thermodynamic equilibrium. No additional suppositions about the shape of its solution or simplifications of the equation itself, that may lower the accuracy of the physical model, are made. Current status of interphase energy, momentum, mass transfer in strong vaporation- condensation of pure vapor is sufficient for calculation of various one-dimensional problems. Two- and three-dimensional problems have been investigated not so comprehensively as for one-dimensional statement. However the investigation of transfer problems namely in many-dimensional statement can have the principal meaning. The paper assumes obtaining the macroparameters field and velocity distribution functions of molecules along the whole flow area. They present independent scientific interest and provide understanding of nonequilibrium gas flows features and revealing the role of molecules interaction in a rarefied environment. Certain attention will be paid to solving of one kinetic equation in the case of pure gas or the kinetic equation system written for different components in the case of gas mixture with calculating of collision integrals considering interaction of particles of different nature.

The method of direct numerical solving of the Boltzmann kinetic equation developed by F.G.Tcheremissine and V.V.Aristov [5]. The direct numerical solving of the Boltzmann equation presupposes introduction of a fixed grid in the velocity and physical space. Transition from constantly changed values to a set of discrete values leads to a system of a large number (of the order of some hundreds or thousands) integro-differential equations. Partial derivates are replaced by their finite difference analogues:

$$\begin{cases} \dfrac{\Delta f_A^k}{\Delta t} + \xi_x^k \dfrac{\Delta f_A^k}{\Delta x} + \xi_y^k \dfrac{\Delta f_A^k}{\Delta y} = J_{AA}^k + J_{AB}^k, \\ \dfrac{\Delta f_B^k}{\Delta t} + \xi_x^k \dfrac{\Delta f_B^k}{\Delta x} + \xi_y^k \dfrac{\Delta f_B^k}{\Delta y} = J_{BB}^k + J_{BA}^k, \end{cases} \tag{3}$$

were k - is number of point in velocity grid.

This system is resolved by an iterative procedure. Calculation of the multidimensional collision integral is made by the Monte-Carlo technique which can be improved by the use of randomized uniformly distributed sequences instead of usual random nodes. The used method is accepted worldwide. It allows one to obtain solving of steady and non-steady many-dimensional problems with complex boundary conditions with sufficient precision and to reveal delicate features of nonequilibrium flow by different potentials of molecules interaction.

3 Results of Computation

Solution results for above described problem were obtained due to the NORMA system application on the system with distributed memory multiprocessers (two Alpha 21264/667MHz in node, memory 1Gb in node, SAN Myrinet to communication, 64 nodes). Norma program was compiled in Fortran with MPI library. Problem size (dimension of grid) is 100 x 100 x 18 x 18 x 9.

For measurements purpose we have used four variants of processors configurations: 20 processors (20 x 1 in line and 5 x 4 in matrix) and 50 processors (50 x 1 in line and 10 x 5 in matrix). Table 1 shows times (in seconds) of computations that were obtained for 100 iterations during solution process.

Table 1. Times (in seconds) of computations for 100 iterations

Processors Time	20(20x1)	20(5x4)	50(50x1)	50(10x5)
Total time	265.61	253.66	142.81	119.88
Communication	30.83	19.19	17.19	10.20

Communication overhead are results of point-to-point communications and all-to-all communications. Volume of all-to-all communications is about 2,2 Mb in each iteration.

We hope, that the approach based on usage of Norma system, can appear useful for creation of mobile effective parallel programs for solution of a practical mathematical physics problems.

Solution results of the problem about gas mixture flow are presented in Fig. 1 - 4. Calculation domain is 50x50 mean free paths of nitrogen molecules at temperature $T_0 = 300$ K and numerical density $n_0 = 2.42 \cdot 10^{17}$. The gas mixture enter in investigated domain through opening and flow out through others boundary surfaces. Maximum value of nitrogen density and helium density in Fig. 1, 2 is $0.50 n_0$.

It should be noted that results were got as solutions of the Boltzmann equation for gas-gas mixture in nonequilibrium problem. The presented pictures are illustrating that flows of mixture components are different. This result is caused by nitrogen and helium molecule interactions.

4 Acknowledgment

The study is supported by Russian Foundation for Basic Researches, Grants 00-02-16273 and 01-01-00411.

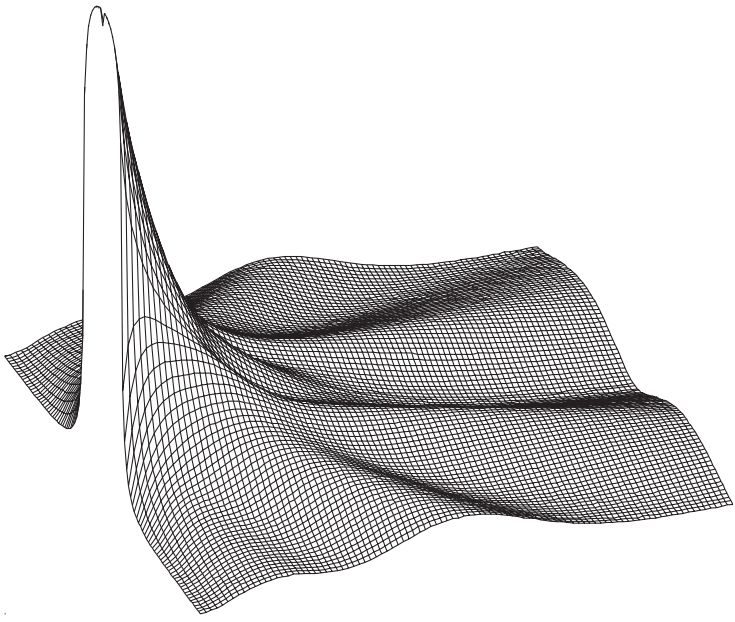


Fig. 1. Nitrogen density

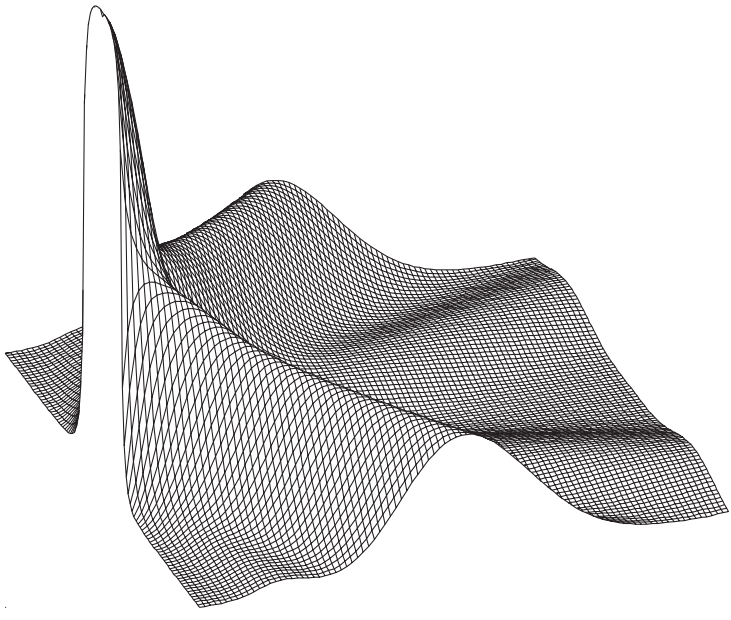


Fig. 2. Helium density

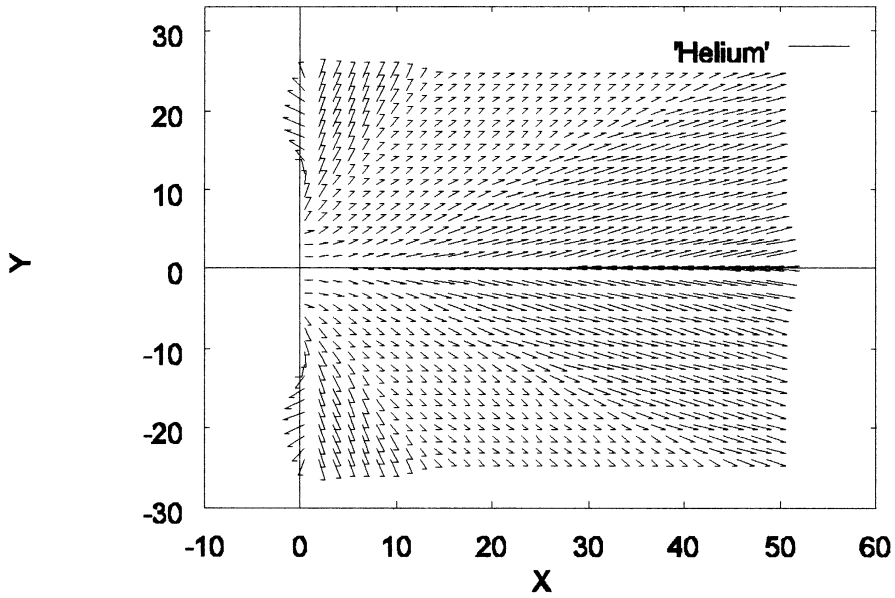


Fig. 3. Nitrogen velocity

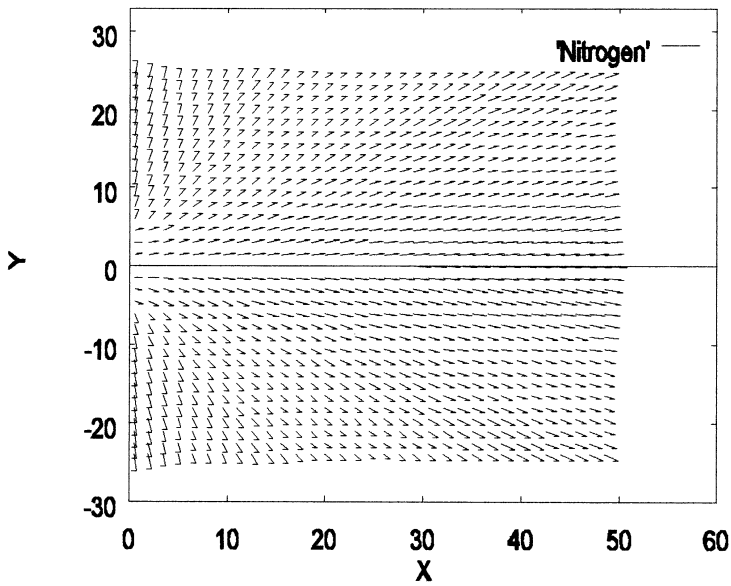


Fig. 4. Helium velocity

References

1. A.N. Andrianov, K.N. Efimkin, I.B. Zadykhailo. Nonprocedural language for mathematical physics. Programming and Computer Software, v17, 2(1991), pp. 121-133
2. A.N. Andrianov, A.B. Bugerja, K.N. Efimkin, I.B. Zadykhailo. The specification of the NORMA language. Preprint of Keldysh Ins. of Appl. Math., Russian Academy of Sc., 120(1995), pp.1-50. (in Russian).
3. I.B. Zadykhailo, K.N. Efimkin. Meaningful terms and new generation languages (the problem of stability, friendly interface and adaptation to execution environment). Information technologies and computational systems, 2(1996), pp. 46-58. (in Russian)
4. A.N.Andrianov. The synthesis of parallel and vector programs by the nonprocedural Norma specification. Ph.D., Moscow, 1990, 131 p. (In Russian).
5. Aristov V.V., Tcheremissine F.G., The direct numerical solving of the kinetic Boltzmann equation., Moscow: Computing Center of the Russian Academy of Sciense. 1992.

Adaptive High-Performance Method for Numerical Simulation of Unsteady Complex Flows with Number of Strong and Weak Discontinuities

Alexander Vinogradov[1], Vladimir Volkov[1], Vladimir Gidaspov[1],

Alexander Muslaev[1], and Peter Rozovski[2]

[1] Moscow State Aviation Institute, Volokolamskoe shosse, 4, Moscow, RUSSIA 125871
[2] Trapezo, 37 Natoma Street, San Francisco, CA 94105 USA
peter@trapezo.com

Abstract. Here we discuss results of further development of numerical method for computer simulation of unsteady flows with number of strong and weak discontinuities in case of nonequilibrium homogeneous condensation. This method is based on idea of explicit tracking of discontinuity surfaces (shock waves, rarefaction waves, contact surfaces) during calculation process. The results for generation of silver clusters in dense vapors during unsteady spherical expansion closed vacuum chamber are presented.

1 Introduction

The solution of many real problems requires numerical treatment of unsteady flows with nonequilibrium processes. Flow field can include strong (shock waves and contact surfaces) and weak (boundary characteristics of expansion waves) discontinuities. As result of interaction of discontinuities their number changes with time. Currently two classes of numerical methods are used for treatment of such flows.

First class consists of shock-capturing methods. They use the same algorithm for all grid points without special procedures for calculation the points on discontinuity surfaces and their main advantage is uniformity. It results in their easy coding. But even best types of them (TVD, ENO, etc.) have disadvantage of spreading discontinuity to 2-3 grid cells. In case of contact surface such spreading is even worse due to unphysical growing of its length with time. It is possible to avoid this problem by using small grid cells near contact surface or special adaptive grids. But it violates the uniformity of algorithm.

Second class of methods includes the methods of tracking all or most intensive discontinuities. Their main principles have been formulated few decades ago and their effectiveness was proved many times. But due to common opinion that discontinuity tracking is extremely complex and poorly formalized process the number of investiga-

V.N. Alexandrov et al. (Eds.): ICCS 2001, LNCS 2073, pp. 511-517, 2001.
© Springer-Verlag Berlin Heidelberg 2001

tions carried out by above methods is very small. In most of them only limited constant number of strong discontinuities is tracked.

The advantage of methods with explicit tracking of discontinuities is simple calculation of hybrid grids with steady regular sub-grid and adaptive moving sub-grid. The last relates to tracked discontinuities and moving boundaries. It results in high-accuracy of calculations even in case of small number of nonmoving grid points.

In our previous publications [1,2] we proposed the numerical technique, which combines the advantages of both classes of methods: uniformity of algorithm and automatic explicit tracking of existed initially and generated later strong/weak discontinuities. This technique was successfully used for simulation of non-reacting gas expansion flow in vacuum chamber with solid [1] and porous [2] walls. Here we present the generalization of this technique for reacting flows with nonequilibrium homogeneous condensation.

2 Mathematical Model of Unsteady Flows of Condensing Vapors

The model is based on quasi-one-dimensional unsteady mass, momentum and energy transfer conservation laws written in differential form in areas of continuity solution and integral form at discontinuity surfaces. Flow is considered as inviscid and with no heat-conduction.

The corresponding system of equations for binary mixture of inert gas and condensing vapor has the next form:

$$\frac{\partial}{\partial t}\begin{bmatrix}\rho F\\ \rho u F\\ \rho e_0 F\\ \rho \gamma_j F\end{bmatrix}+\frac{\partial}{\partial r}\begin{bmatrix}\rho u F\\ (\rho u^2+p)F\\ \rho u(e_0+p\rho^{-1})F\\ \rho u \gamma_j F\end{bmatrix}=\begin{bmatrix}0\\ p\dfrac{dF}{dr}\\ 0\\ \rho W_j F\end{bmatrix} \tag{1}$$

Where t is time, r is linear coordinate, $e_0 = e + u^2/2, \rho, u, p, e$ are density, velocity, pressure and internal energy of mixture, $F = r^\nu$, ν=0,1,2 for plane, cylindrical and spherical symmetry, γ_j are mole-mass concentrations of clusters with j atoms, W_j are rates of concentration change as a result of nonequilibrium homogeneous condensation. In last equation index j varies from 1 to ∞. The special transformation is used to approximate infinite system of kinetic equations in (1) by finite system of N equations [3].

The system (1) is completed by state equations

$$e = \gamma_A e_A(T) + \sum_{j=1}^{N} \gamma_j e_j(T), \rho = \frac{p}{RT}\left[\gamma_A + \sum_{j=1}^{N} \gamma_j\right]^{-1} \quad (2)$$

And condition of constant number of atoms

$$\gamma_A = const, \sum_{j=1}^{N} j\gamma_j = const, \quad (3)$$

Where T is temperature, γ_A is mole-mass concentration of inert gas, R is universal gas constant.

The formulae and methods for calculation $W_j, e_A(T), e_j(T)$ are presented in [3], there also we discuss the methods of selection the appropriate value for N.

3 Numerical Method

The suggested numerical method is based on the next principles:

- the flow is calculated by grid-characteristics method by t=const layers,
- the whole flow region is divided to sub-regions with continuity parameters in them, and their boundaries are surfaces of strong/weak discontinuities or external left/right boundaries of the region,
- numerical grid consists from non-moving and moving points. The last ones are points of discontinuities. Pairs of values corresponding left and right limits of continuum solution describe parameters in them. If discontinuity is stable (it has the same type on the next time step), the left/right values satisfy to integral relations for this type of discontinuity,
- for crossing points of two strong discontinuities the left/right values don't satisfy to such relation. The formation of new discontinuity surfaces occurs and determination of number, types and parameters of these discontinuities should be obtained from solution of corresponding Riemann problem;
- the standard time step is selected from the Courant condition. But if crossing of any discontinuities happens before next time, the exact time of nearest crossing is taken;
- the calculation process is divided into gas-dynamic half-step (calculation of u, p, T from calculated previously concentrations and right-hand-sides of kinetic equations) and kinetic half-step (calculation of concentrations by integration of kinetic equations);
- the possibility of generation of shock wave in initially continuous sub-region is analyzed. The coordinate and time of such generation point is calculated from crossing point of characteristics of the same type;
- flow parameters in each of grid nodes are calculated from characteristic conditions. The base points can locate as on previous time layer as on discontinuities surfaces. That means that all base points relates to one sub-region of

continuum solution. The resulting system of equations is solved numerically and if there is no convergence for given number of iterations, the time step is decreased.

Any grid points relates to one of the following types: symmetry axis, solid wall, shock wave, contact surface, discontinuity characteristic, usual characteristic, trajectory and nonmoving usual grid point. From physical consideration it is clear that 18 types of crossing can take place.

For interactions shock wave/shock wave or shock wave/contact surface the structure and parameters of resulting flow are calculated from generalized solution of Riemann problem on case of reacting gases with variable ratios of specific heats. It can be shown that for thermodynamic model of inert gas/condensing vapor mixture [3] this problem has unique solution.

After solution of Riemann problem initial parameters of all generated discontinuities are known. For extra-intensive expansion waves the several additional usual characteristics are put into wave. Their number is proportional to pressure gradient in expansion wave. All generated discontinuities have the same coordinate but different parameters.

For shock wave/solid wall or shock wave/symmetry axis the parameters of reflected shock are calculated from Rankine-Hugoniot type conditions for multicomponent gas and condition u=0 behind the shock.

In case of flow with cylindrical or spherical symmetry near symmetry axis (r=0) the intensity of shock wave is infinite. The point of shock wave/symmetry axis interaction is special point with infinite values of shock velocity and pressure behind shock. The flow in such case can be calculated from known solution of Huderlay-Landau-Stanukovich by special algorithm.

The interactions of usual characteristics with usual characteristic, discontinuity characteristic, contact surface, solid wall, shock wave, symmetry axis and discontinuity characteristic with contact surface, shock wave, discontinuity characteristic, solid wall, symmetry axis are limiting cases of above situations. Further tracking of given characteristic or its exchange to characteristic of another family at interaction with solid wall or contact surface is determined from flow properties.

Parameters in points of interactions of trajectories with usual or discontinuity characteristics are calculated by usual method of characteristics. For trajectory/solid wall interaction parameters on trajectory are taken from point behind shock.

The main advantage of proposed method is avoiding need of a priory knowledge of flow structure. As it can be seen from presented far results, in some cases it is principally impossible due to extra-complexity of flow. The formal character of algorithm which consists from standard steps of solution of standards problems allow one to calculate flow region uniformly without changing program logic for new class of flows. The researcher needs only to know the coordinates, types and parameters of discontinuities on initial time layer. So the suggested method has all advantages of "shock-capturing" numerical schemes but allow obtain exact flow picture.

Of course as in any numerical scheme there are some "fitting" parameters like number of usual characteristics in extensive expansion waves, which should be known before calculation. Also in some cases the number of tracking discontinuities begins to grow seriously. That results in computer memory leak. In real calculations the special

"destruction" procedure for tracked weak (strong) discontinuities with small difference between left and right values of gradients (parameters) is applied. Their "destruction" condition is the error of integral conservation laws.

After determination of parameters of discontinuities on the next time step the calculation of inner points in each region doesn't depends from each other. So the proposed method can be easily adopted for parallel execution.

For nonequilibrium calculations for solution of kinetic equations takes the main part of execution time. As it was shown in [4] it is possible to improve performance by redesign this part of program code according to vectorized computer ELECTRONIKA–SSBIS.

4 Gas Cloud Expansion into Closed Vacuum Chamber

The features of above technique are demonstrated on case of unsteady expansion of gas cloud of binary mixture of inert gas (argon) and condensing vapor (silver) into closed vacuum chamber filled by residual gas (air). Such flow is realized at shock tubes with chambers of high/low pressure, separated by membrane, at expansion of vapors generated by electrical explosion of thin wire in cylindrical vacuum tube, at expansion flow of substance evaporated by laser beam. The similar flows 6are considered in some astrophysical problems.

Due to high initial values of initial temperature and cooling rate the nonequilibrium homogeneous condensation can take place in such flow.

The initial cloud radii equals to 1 mm, the chamber's radii equals to 5 mm. The initial pressure and temperature of silver vapor equal to 5270 Pa and 2000 K. The pressure and temperature of residual gas equal to 10 Pa and 300 K, respectively. At Fig. 1 the r-t diagram for this flow is presented. One can see head shock wave, running at the low-pressure region, contact surface and centered rarefaction wave. Later head shock wave reflects from solid wall and, after interaction with contact surface, transforms into two shock waves moving in opposite directions. The left shock wave moves to the symmetry axis, reflects from it and comes back to solid wall. The right shock wave interacts with solid wall and this process repeats many times. The resulting flow has complex structure with lot of strong/weak discontinuities. As it is seen at Fig.1 that suggested numerical technique permits to receive "exact" picture of flow with all details.

The parameterized simulation of flows for various ratios z of chamber/cloud initial sizes and for various types of symmetry shows that for large z values the flow picture at initial times is simpler (less number of shock waves). The reason is that reflection of shocks from symmetry axis requires more time. Also there is difference of temperature behavior at solid wall. For z=1/5 after increase of temperature at head shock wave it begins to decrease significantly. In case of z=1/5 the decreasing time is small and later temperature has stable level.

Against plane case in cylindrical/spherical flows the ending right characteristic of rarefaction wave transforms into shock wave as a result of interactions of characteristics of same family. The intensity of generated shock wave is greater in spherical case.

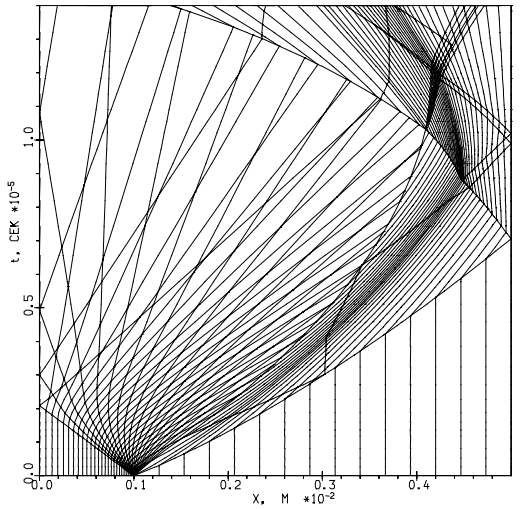


Fig. 1. The r-t diagram of unsteady expansion flow (z=1/5).

At Fig.2 the pressure-temperature diagram for first three trajectories behind contact surface is present. The results agree with usual pictures for nonequilibrium homogeneous condensation in high-speed flows. Initially the active changes of cluster size distribution function take place. But due to small values of cluster concentrations the mixture parameters vary according to adiabatic law. After formation of enough number of clusters (Wilson point) their fast growth step begins resulting in decreasing of super-saturation ratio. It is seen that maximal super-cooling value decreases with increasing of initial distance between trajectories and contact surface. That agrees with known data about nonequilibrium condensation.

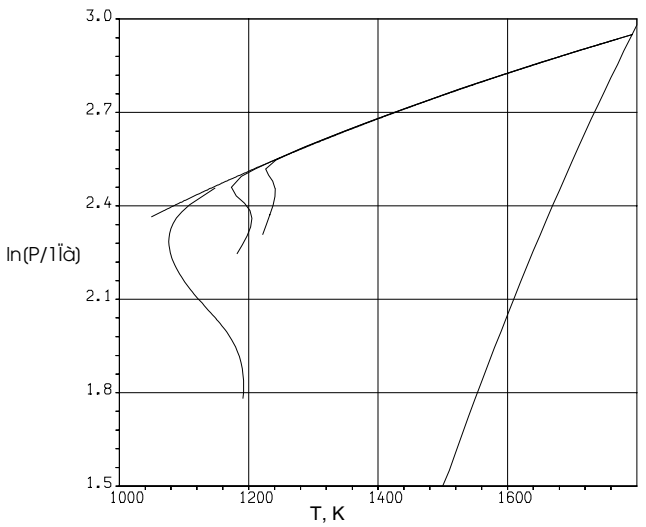


Fig. 2. The P-T diagram for unsteady expansion flow (z=1/5).

5 Conclusion

The suggested numerical technique can be used for wide class of 1D unsteady reacting flows with chock waves, detonation waves and other strong/weak discontinuities. The main advantages of above technique can be realized in case of reacting flows with unknown structure and without reliable kinetic mechanisms due to possibility to obtain all flow details.

References

1. Vinogradov A.V., Volkov V.A., Gidaspov V.Yu., Rozovski P.V. : Influence of Residual Gas on the Expansion of a Dence Gas Cloud in a Vacuum Chamber and itsInteraction with a Target or Wall. Technical Physics. 38 (1993) 946-948
2. Vinogradov A.V., Volkov V.A., Gidaspov V.Yu., Rozovski P.V. : Interaction of an Expanding Gas Cloud with a Perforated Screen. . Technical Physics. 42 (1997) 473-476
3. Volkov V.A., Muslaev A.V., Pirumov U.G., Rozovski P.V.: Nonequilibrium Condensation of Metal Vapor Mixed with Inert Gas in Nozzle Expansion in Cluster Beam Generator. Fluid Dynamics. 30 (1995) 335-486
4. Marasanov A.M., Rozovski P.V., Shebeko Yu.A. : Development of Software Package for Physical Gas Dynamics Problems for Vectorized Computers. Computer Technologies. 2 (1992) 223-231 (in Russian)

Cellular Automata as a Mesoscopic Approach to Model and Simulate Complex Systems

P.M.A. Sloot and A.G. Hoekstra

Section Computational Science, Faculty of Science, University of Amsterdam,
Kruislaan 403, 1098 SJ Amsterdam, The Netherlands
{sloot, alfons}@science.uva.nl

Abstract. We discuss the cellular automata approach and its extensions, the lattice Boltzmann and multiparticle methods. The potential of these techniques is demonstrated in the case of modeling complex systems. In particular, we consider simple applications taken from various scientific domains. We discuss our distributed particle simulation of flow, based on a parallel lattice Boltzmann Method. Efficient parallel execution is possible, provided that (dynamic) load balancing techniques are applied. Next, we present a number of case studies of flow in complex geometry, i.e. flow in porous media and in static mixer reactors.

1. The Cellular Automata Approach

A natural way to describe a physical, chemical or biological system is to propose a model of what we think is happening. During this process we try to keep only the ingredients we believe to be essential whilst still capturing the behavior we are interested in. Using an appropriate mathematical machinery, such a model can be expressed in terms a set of equations whose solution gives the desired answers on the system. The description in terms of equations is very powerful and corresponds to a rather high level of abstraction. For a long time, this methodology has been the only tractable way for scientists to address a problem.

Another approach, which has been made possible by the advent of fast computers, is to stay at the level of the model and its basic components. The idea is that all the information is already contained in the model and that a computer simulation will be able to answer any possible question on the system by just running the model for some time. Thus, there is no need to use a complicated mathematical tool to obtain a high level of description. We just need to express the model in a way which is suitable to an effective computer implementation. Cellular automata constitute a paradigm in which simple models of complex phenomena can be easily formulated. In particular, cellular automata models illustrate the fact that a complex behavior emerges out of many simply interacting components through a collective effect.

The degree of reality of the model depends on the level of description we expect. When we are interested in the global or macroscopic properties of a system the microscopic details of a system are often irrelevant. On the other hand, symmetries and conservation laws are usually the essential ingredients of such mesoscopic

V.N. Alexandrov et al. (Eds.): ICCS 2001, LNCS 2073, pp. 518-527, 2001.
© Springer-Verlag Berlin Heidelberg 2001

description. It is therefore a clear advantage to invent a much simpler microscopic reality, which is more appropriate to our computational means of investigation.

A cellular automata model can be seen as a fictitious universe which has its own microscopic reality but, nevertheless, has the same macroscopic behavior as the real system we are interested in. The examples in the next section will illustrate this statement.

Cellular automata (CA) are an idealization of a physical system in which space and time are discrete. In addition, the physical quantities (or state of the automaton) take only a finite set of values. Since the time it has been invented by von Neumann in the late 1940s, the cellular automata approach has been applied to a large range of scientific problems [1, 2].

The original motivation of von Neumann was to extract the abstract mechanisms leading to self-reproduction of biological organisms. [2]

Following the suggestions of S. Ulam, von Neumann addressed this question in the framework of a fully discrete universe made up of cells. Each cell is characterized by an internal state, which typically consists of a finite number of information bits. Von Neumann suggested that this system of cells evolves, in discrete time steps. The rule, determining the evolution of this system is the same for all cells and is a function of the states of the neighboring cells. Similarly to what happens in any biological system, the activity of the cells takes place simultaneously. However, the same clock drives the evolution of each cell and the updating of the internal state of each cell occurs synchronously. Such fully discrete dynamical systems (cellular space) as invented by von Neumann are now referred to as a *cellular automata* (CA).

After the work of von Neumann, other authors have followed the same line of research and nowadays the problem is still of interest [3] and has lead to interesting developments for new computer architectures and algorithms. [4, 5]

A very important feature of CAs is that they provide simple models of complex systems. They exemplify the fact that a collective behavior can emerge out of the sum of many, simply interacting, components. Even if the basic and local interactions are perfectly known, it is possible that the global behavior obeys new laws that are not obviously extrapolated from the individual properties, as if the whole is more than the sum of all the parts. This properties makes cellular automata a very interesting approach to model physical systems and in particular to simulate complex and nonequilibrium phenomena. [6]

The studies undertaken by S. Wolfram in the 1980s [7] clearly establishes that a CA (the famous Wolfram's rules) may exhibits many of the behaviors encountered in continuous dynamical systems, yet in a much simpler mathematical framework. A further step is to recognize that CAs are not only behaving similarly to some dynamical processes, they can also represent an actual model of a given physical system, leading to macroscopic predictions that could be checked experimentally. This fact follows from statistical mechanics which tells us that the macroscopic behavior of many systems is quite disconnected from its microscopic reality and that only symmetries and conservation laws survives to the change of observation level: it is well known that the flows of a fluid, a gas or even a granular media are very similar at a macroscopic scale, in spite of their different microscopic nature.

Although John von Neumann introduced the cellular automata theory several decades ago, only in recent years it became significant as a method for modeling and simulation of complex systems. This occurred due to the implementation of cellular automata on massively parallel computers. Based on the inherent parallelism of

cellular automata, these new architectures made possible the design and development of high-performance software environments. These environments exploit the inherent parallelism of the CA model for efficient simulation of complex systems modeled by a large number of simple elements with local interactions. By means of these environments, cellular automata have been used recently to solve complex problems in many fields of science, engineering, computer science, and economy. In particular parallel cellular automata models are successfully used in fluid dynamics, molecular dynamics, biology, genetics, chemistry, road traffic flow, cryptography, image processing, environmental modeling, and finance. [3]

Interesting examples are the Lattice Gas Automata for fluid flow, and the derived Lattice Boltzmann models. The sequel of the paper will introduce these mesoscopic models and demonstrate the potential of Cellular Automata as a Mesoscopic Approach to Model and Simulate Complex Systems.

2. Particle Simulation of Flow

Some 12 years ago theoretical physicists showed that an highly idealized model of a gas, consisting of particles that have a very limited set of velocities and that are confined to a lattice behaves, on average, as an incompressible fluid [8]. This Lattice Gas Automaton (LGA) is a particle model for fluid flow and is a true CA. In this section we shortly introduce LGA and show how it relates to CA. Detailed theory behind LGA is introduced in two recent books. [9, 10]

Consider a hexagonal lattice, as in Fig. 1. Particles live on the links of the lattice, and they can move from node to node. The dynamics of the particles is such that they all move from one node to another. Next, if particles meet on a node, they collide and change direction (see Fig. 1). The collisions are such that they obey the physical constraints of conservation of mass, momentum, and energy.

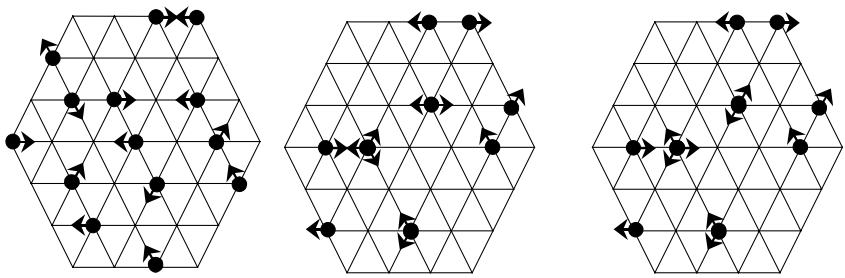


Fig. 1. Lattice and particle update mechanism for a LGA. A dot denotes a particle and the arrow its moving direction. From left to right an initial condition, streaming, and collision of particles are shown.

We can formally define a CA rule for such an LGA as follows. Suppose that the state of a cell is determined by b_m surrounding cells. Usually, only the nearest and next-nearest neighbors are considered. For example, on a square lattice with only nearest neighbor interactions $b_m = 4$, if next-nearest neighbors are also included $b_m = 8$, and on a hexagonal lattice with nearest neighbor interactions $b_m = 6$. Furthermore,

suppose that the state of the cell is a vector **n** of $b = b_m$ bits. Each element of the state vector is associated with a direction on the CA lattice. For example, in the case of a square grid with only nearest neighbor interactions we may associate the first element of the state vector with the *north* direction, the second with *east*, the third with *south*, and the fourth with *west*. With these definitions we construct the following CA rule (called the LGA rule), which consists of two sequential steps:

1. Each bit in the state vector is moved in its associated direction (so in the example, the bit in element 1 is moved to the neighboring cell in the north) and placed in the state vector of the associated neighboring cell, in the same position (so, the bit in element 1 is moved to element 1 of the state vector in the cell in the north direction). In this step each cell is in fact moving bits from its state vector in all directions, and at the same time is receiving bits from all directions, which are stored into the state vector.
2. Following some deterministic or stochastic procedure, the bits in the state vector are reshuffled. For instance, the state vector (1,0,1,0) is changed to (0,1,0,1).

Maybe very surprisingly, if we assign physical quantities to this CA, enforce physical conservation laws on the bit reshuffling rule of step 2, and use methods from theoretical physics to study the dynamics, we are in fact able to analyze the CA in terms of its average behavior. The average state vector of a cell and the average flow of bits between cells can be calculated. Even better, it turns out, again within the correct physical picture that this CA behaves like a real fluid (such as water) and therefore can be used as a model for hydrodynamics. Furthermore, as the LGA rule is intrinsically local (only nearest and next nearest neighbor interactions) we constructed an inherently parallel particle model for fluid flow.

Associate the bits in the state vector with *particles*; a one-bit codes for the presence of a particle, and a zero bit codes for the absence of a particle. Assume that all particles are equal and have a mass of 1. Step 1 in the LGA-CA is now interpreted as streaming of particles from one cell to another. If we also introduce a length scale, i.e. a *distance* between the cells (usually the distance between nearest neighbors cells is taken as 1) and a time scale, i.e. a *duration* of the streaming (i.e. step 1 in the LGA-CA rule, usually a time step of 1 is assumed), then we are able to define a *velocity* $\mathbf{c}_i$ for each particle in direction i (i.e. the direction associated with the i-th element of the state vector **n**). Step 1 of the LGA-CA is the streaming of particles with velocity $\mathbf{c}_i$ from one cell to a neighboring cell. Now we may imagine, as the particles meet in a cell that they collide. In this collision the velocity of the particles (i.e. both absolute speed and direction) can be changed. The reshuffling of bits in step 2 of the LGA-CA rule can be interpreted as a collision of particles. In a real physical collision, mass, momentum, and energy are conserved. Therefore, if we formulate the reshuffling such that these three conservation laws are obeyed, we have constructed a true Lattice Gas Automaton, i.e. a gas of particles that can have a small set of discrete velocities $\mathbf{c}_i$, moving in lock-step over the links of a lattice (space is discretized) and that all collide with other particles arriving at a lattice point at the same time. In the collisions, particles may be sent in other directions, in such a way that the total mass and momentum in a lattice point is conserved.

We can now define for each cell of the LGA-CA a density ρ and momentum $\rho\mathbf{u}$, with **u** the velocity of the gas:

$$\rho = \sum_{i=1}^{b} N_i \,, \quad \rho \mathbf{u} = \sum_{i=1}^{b} \mathbf{c}_i N_i \,, \tag{1}$$

where $N_i = \langle \mathbf{n}_i \rangle$, i.e. a statistical average of the Boolean variables; N_i should be interpreted as a particle density.

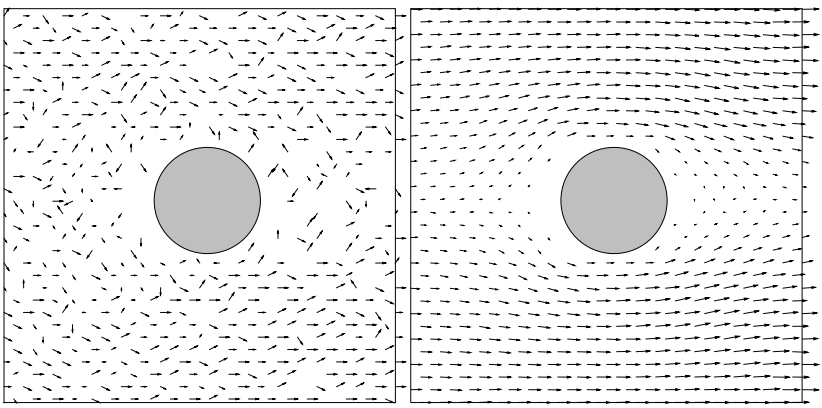


Fig. 2. LGA simulation of flow around a cylinder. The arrows are the flow velocities, the length is proportional to the absolute velocity. The simulations were done with FHP-III, on a 32×64 lattice, the cylinder has a diameter of 8 lattice spacings, only a 32×32 portion of the lattice is shown; periodic boundary conditions in all directions are assumed. The left figure shows the result of a single iteration of the LGA, the right figure shows the velocities after averaging over 1000 LGA iterations.

If we let the LGA evaluate and calculate the density and momentum as defined in eqn. (1), these quantities behave just like a real fluid. In Fig. 2 we show an example of an LGA simulation of flow around a cylinder. In left figure we show the results of a single iteration of the LGA, so in fact we have assumed that $N_i = \mathbf{n}_i$. Clearly, the resulting flow field is very noisy. In order to arrive at smooth flow lines one should calculate $N_i = \langle \mathbf{n}_i \rangle$. Because the flow is static, we calculate N_i by averaging the Boolean variables $\mathbf{n}_i$ over a large number of LGA iterations. The resulting flow velocities are shown in the left panel of Fig. 2.

Immediately after the discovery of LGA as a model for hydrodynamics, it was criticized on three points; noisy dynamics, lack of Galilean invariance, and exponential complexity of the collision operator. The noisy dynamics is clearly illustrated in Fig. 2. The lack of Galilean invariance is a somewhat technical matter which results in small differences between the equation for conservation of momentum for LGA and real Navier-Stokes equations, for details see e.g. [10]. Finally, adding more velocities in an LGA leads to increasingly more complex collision operators, exponentially in the number of particles. Therefore, another model, the Lattice Boltzmann Method (LBM), was introduced. This method is reviewed in detail in [11].

The basic idea in LBM is that one should not model the individual particles $\mathbf{n}_i$, but immediately the particle densities N_i. This means that particle densities are streamed

from cell to cell, and particle densities collide. This immediately solves the problem of noisy dynamics. However, in a strict sense we no longer have a CA with a Boolean state vector. However, we can view LBM as a generalized CA. It is easy to make LBM Galilean invariant, thus solving the second problem of LGA. Finally, a very simple collision operator is introduced. This so-called BGK collision operator models the collisions as a single-time relaxation towards equilibrium. This L-BGK method is also developed for many other lattices, e.g. in two or three-dimensional cubic lattices with nearest and next nearest neighbor interactions. The LBM and especially the L-BGK has found widespread use in simulations of fluid flow.

3. Parallel Lattice Gas and Lattice Boltzmann Simulations

The local nature of the LGA and LBM interactions allows a very straightforward realization of parallelism. A geometric decomposition of the lattice with only local message passing between the boundaries of the different domains is sufficient to realize an efficient parallel simulation. For instance, we have developed a generic 2-dimensional LGA implementation that is suitable for any multi-species (thermal) LGA [12]. Parallelism was introduced by means of a 1-dimensional, i.e strip-wise, decomposition of the lattice. As long as the grid dimension compared to the number of processors is large enough, this approach results in a very efficient parallel execution.

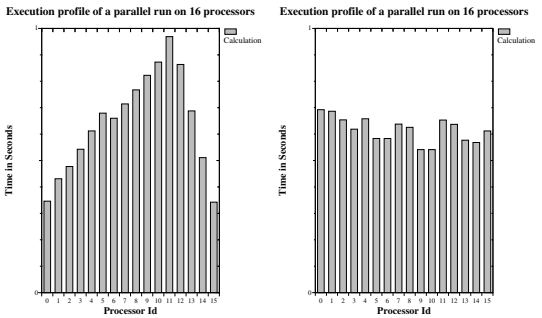


Fig. 3. The processor load, on 16 processors, for slice and ORB decomposition of the elutriator chamber benchmark (see [13]). On the x-axis are the processor ID's and on the y-axis the execution time of each processor.

This LGA system is mainly used for simulations in simple rectangular domains without internal obstacles. However, in a more general case, where the boundaries of the grid have other forms and internal structure (i.e. solid parts where no particles will flow) the simple strip-wise decomposition results in severe load imbalance. In this case, as was shown in [13], a more advanced decomposition scheme, the Orthogonal Recursive Bisection (ORB) method [14], still leads to highly efficient parallel LBM simulations. ORB restores the load balancing again, however at the price of a somewhat more complicated communication pattern between processors. In Fig. 3 we

show, for a representative 2D benchmark, the processor load. In a simple slice decomposition, the load on each processor differs a lot. For the ORB the load is almost balanced. This results for this benchmark, in reductions of execution times as large as 40%. For details we refer to [13].

For flow problems with a static geometry the ORB decomposition seems to be most appropriate. However, we are also interested in flow simulations where the geometry dynamically changes during the simulation. Here we refer to e.g. free flow around growing biological objects [15] or bounded flows with dynamically changing boundaries (e.g. blood flow in the heart). In this case an initially well-balanced parallel computation may become highly unbalanced. This may be overcome by a redundant scattered decomposition or by dynamic load balancing. Here we report some preliminary results of the first approach, the latter will be published elsewhere [16].

We consider growth of an aggregate in a three dimensional box. We must simulate flow around the object. The details of the streamlines close to the object determine its local growth. Due to the growth of the aggregate a straightforward decomposition (for example partitioning of the lattice in equal sized slices or boxes) would lead to strong load imbalance. To solve this problem we have tested two strategies to obtain a more equal distribution of the load over the processors:

1. Box decomposition in combination with scattered decomposition.
2. Orthogonal Recursive Bisection (ORB) in combination with scattered decomposition.

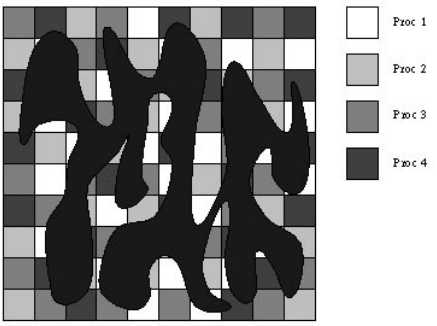


Fig. 4. Decomposition of an irregular shaped object in a 2D lattice. In this case 100 blocks are scattered over 4 processors.

The idea is to decompose the grid in a large number of partitions, much larger than the number of available processors, using either a box decomposition or ORB. Next, the partitions are randomly assigned to a processor. An example of a scattered decomposition over 4 processors of an irregular shaped object in a 2D lattice is shown in Fig. 4. In this example the lattice is divided into 100 blocks, where each block is randomly assigned to one of the four processors. Most of the computation is done in the blocks containing exclusively fluid nodes. The scattering of the blocks over the processors leads to a spatial averaging of the load, where decreasing block sizes cause a better load balancing, but also an increasing communication overhead. [17] Especially in simulations in which the shape of the object cannot be predicted,

redundant scattered decomposition is an attractive option to improve the load balance in parallel simulations [18, 19].

We have compared both decomposition strategies by computing the load balancing efficiency:

$$\varepsilon_{load} = l_{\min} / l_{\max} , \tag{2}$$

where $l_{\min}$ is the load of the fastest process and $l_{\max}$ the load of the slowest process. The two decomposition strategies were tested by using two extreme morphologies of the aggregates, a very compact shape and a dendritic shaped aggregate, and by measuring the load balancing efficiency during one iteration of the parallel LBM on 4 processors. The results are shown in Fig. 5. These results indicate that the combination of redundant ORB or box decomposition in combination with a scattered decomposition of partitions on processors may indeed improve load balancing. However, a disadvantage is that although the load balancing efficiencies increase with the number of redundant blocks, the communication overhead also increases. Furthermore, this test was for a single iteration only. We are currently working on testing these methods for real growing objects.

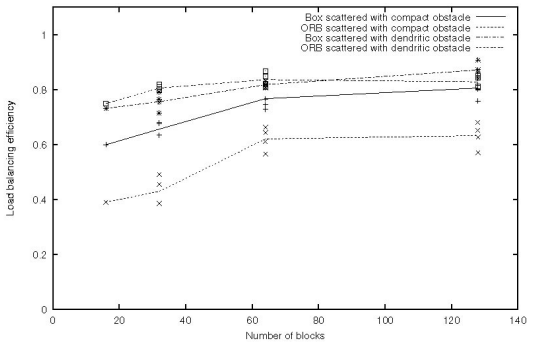


Fig. 5. The load balancing efficiency as a function of the total number of redundant blocks scattered on 4 processors, for box and ORB decomposition and for a compact and dendritic aggregate.

4. Cases: Flow in Complex Geometry

We applied our distributed particle simulation environment as described above for a large number of simulations of fluid flow in complex geometries. Here we show two representative examples, that of flow in random fiber networks and of flow in a static mixer reactor.

The first example is flow in a random fiber network, as drawn in Fig. 6. The fiber network is a realistic model of paper, and the question was to obtain the permeability of the network as a function of the volume fraction of fibers. Simulations were performed on 32 nodes of a Cray T3E using our parallel LBM environment described in the previous section. We obtained permeability curves that are in very good agreement with experimental results (see [20]).

Another impressive example is flow in a Static Mixer Reactor [21]. In such a mixer high viscosity fluids are mixed by letting them flow around a complex arrangement of internal tubes. A typical mixer is shown in Fig. 7. Here, LBM simulations and conventional Finite Element simulations where compared, which agreed very well. The simulation results also agree very well with experimental results. This shows that LBM, which is much easier to parallelize and much easier to extend with more complex modeling compared to Finite Element, (multi-species flow, thermal effects, reactions), is very suitable in real life problems involving complex flow.

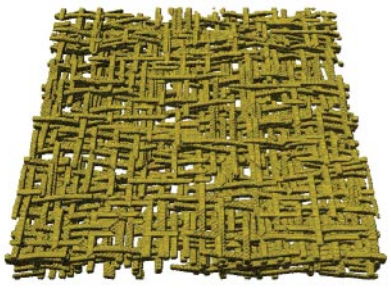


Fig. 6. A random fiber network.

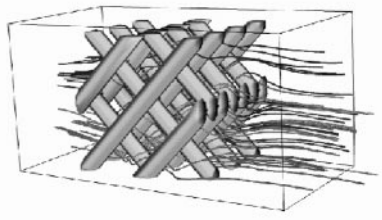


Fig. 7. A Static Mixer Reactor. Flow lines resulting from LBM simulations are also shown.

5. Conclusions

Within the general concept of Cellular Automata we have developed a distributed mesoscopic particle simulation environment for fluid flow. Parallel Lattice Gas Automata and Lattice Boltzmann methods have been realized, and we showed that by carefully taking load balancing into account it is possible to achieve very high parallel efficiencies. By means of two realistic examples of flow in complex geometries the power and potencies of such a distributed particle simulation environment were demonstrated.

References

1. Toffoli, T.,Margolus, N.: Cellular Automata Machines - A New Environment for Modelling. MIT Press, Cambridge, MA (1987)
2. von Neumann, J.: Theory of Self-Reproducing Automata. In: A.W. Burks (eds.): University of Illinois Press, Champaign, Il (1966)
3. Talia, D.,Sloot, P. (eds.):Special Issue: Cellular Automata: Promises and Prospects in Computational Science. Future Generation Computer Systems, (1999)
4. Sloot, P.M.A., Kaandorp, J.A., Hoekstra, A.G.,Overeinder, B.J.: Distributed Cellular Automata : Large Scale Simulation of Natural Phenomena. In: A. Zomaya (eds.): Solutions to Parallel and Distributed Computing Problems: Lessons from Biological Sciences. John Wiley & Sons, (2001)

5. Mange, D.,Tomassini, M. (eds.):Bio-inspired Computing Machines. Lausanne (1998)
6. Moore, C.,Nordahl, M.G.: Lattice Gas Prediction is P-Complete. Santa Fe Institute, 97-04-034 (1997)
7. Wolfram, S.: Cellular Automata and Complexity. Addison-Wesley, (1994)
8. Frish, U., Hasslacher, B.,Pomeau, Y.: Lattice-gas automata for the Navier-Stokes equation. Phys. Rev. Lett. **56** (1986) 1505
9. Chopard, B.,Droz, M.: Cellular Automata Modelling of Physical Systems. Cambridge University Press, (1998)
10. Rothman, D.H.,Zaleski, S.: Lattice-Gas Cellular Automata, Simple Models of Complex Hydrodynamics. Cambridge University Press, Cambridge (1997)
11. Chen, S.,Doolen, G.D.: Lattice Boltzmann Method for Fluid Flows. Ann. Rev. Fluid Mech. **30** (1998) 329
12. Dubbeldam, D., Hoekstra, A.G.,Sloot, P.M.A.: Computational Aspects of Multi-Species Lattice-Gas Automata. In: P.M.A. Sloot, Bubak, M., Hoekstra, A.G. & Hertzberger, L.O. (eds.): Proceedings of the International Conference HPCN Europe '99. Springer Verlag, Berlin (1999) 339-349
13. Kandhai, D., Koponen, A., Hoekstra, A.G., Kataja, M., Timonen, J.,Sloot, P.M.A.: Lattice Boltzmann Hydrodynamics on Parallel Systems. Comp. Phys. Comm. **111** (1998) 14-26
14. Simon, H.D.: Partioning of unstructured problems for parallel processing. Computing Systems in Engeneering **2** (1991) 135-148
15. Kaandorp, J.A., Lowe, C., Frenkel, D.,Sloot, P.M.A.: The effect of nutrient diffusion and flow on coral morphology. Phys. Rev. Lett. **77** (1996) 2328-2331
16. Schoneveld, A.,de Ronde, J., accepted for publication in Fut. Gen. Comp. Syst., 1999.
17. de Ronde, J., Schoneveld, A.,Sloot, P.M.A.: Load balancing by redundant decomposition and mapping. In: H. Liddell, Colbrook, A., Hertzberger, B. & Sloot, P. (eds.): High Performance Computing and Networking (HPCN'96). (1996) 555-561
18. Machta, J.,Greenlaw, R.: The parallel complexity of growth models. Journal of Statistical Physics **77** (1994) 755-781
19. Machta, J.: The computational complexity of pattern formation. Journal of Statistical Physics **70** (1993) 949-967
20. Koponen, A., Kandhai, D., Hellin, E., Alava, M., Hoekstra, A., Kataja, M., Niskanen, K., Sloot, P.,Timonen, J.: Permeability of three-dimensional random fiber webs. Phys. Rev. Lett. **80** (1998) 716-719
21. Kandhai, D., Vidal, D., Hoekstra, A., Hoefsloot, H., Iedema, P.,Sloot, P.: Lattice-Boltzmann and Finite-Element Simulations of Fluid Flow in a SMRX Static Mixer. Int. J. Num. Meth. Fluids **31** (1999) 1019-1033

Ab-Initio Kinetics of Heterogeneous Catalysis: NO + N + O/Rh(111)

A.P.J. Jansen, C.G.M. Hermse, F. Frechard, and J.J. Lukkien

Schuit Institute of Catalysis,
ST/SKA, Eindhoven University of Technology,
P. O. Box 513,
5600 MB Eindhoven,
The Netherlands,
tgtatj@chem.tue.nl,
WWW home page: http://www.catalysis.nl/theory/

Abstract. We show that advances in two fields of computational chemistry, Dynamic Monte Carlo simulations and Density-Functional Theory calculations, are now making it possible to do *ab-initio* kinetics of realistic surface reactions. We present results of simulations of Temperature- Programmed Desorption experiments of NO reduction to N_2 and O_2 on the Rh(111) surface. Kinetic parameters were obtained from Density-Functional Theory calculations with the Generalized Gradient Approximation, making this one of the first, and up till now the most complex, example of *ab-initio* kinetics in heterogeneous catalysis. Top, hcp, and fcc sites are all involved and also lateral interactions are necessary to understand the kinetics of this system.

1 Introduction

Although kinetics plays such an important role in catalysis, its theory has for a long time mainly been restricted to macroscopic rate equations. These implicitly assume a random distribution of adsorbates on a catalyst's surface. Effects of interactions between adsorbates (lateral interactions), reactant segregation, site blocking, and defects have only been described *ad hoc*. With the advent of Dynamic Monte-Carlo (DMC) simulations, also called Kinetic Monte-Carlo simulations, it has only recently become possible to follow the kinetics of reaction systems on an atomic scale, and thus to study these effects properly.

Two developments have played a major role in the advance of DMC. One is the derivation of a Master Equation (ME) on which DMC can be based, and which has kinetic parameters that can be calculated with *ab-initio* quantum chemical methods.[1] This ME links the processes on an atomic scale to the macroscopic kinetics. The other is the development of new and the improvement of existing DMC algorithms and ways of modeling reaction systems,[2,3,4] and the implementation of the algorithms in the general-purpose code CARLOS that allows a user to simulate almost any system of surface reactions.[5] Lateral interactions, steps, and other defects can now easily be included, whereas reactant segregation and site blocking can often be seen in simulations as a consequence of the reaction model.

V.N. Alexandrov et al. (Eds.): ICCS 2001, LNCS 2073, pp. 531–540, 2001.
© Springer-Verlag Berlin Heidelberg 2001

DMC simulations need as input the kinetic parameters of the ME. For simulations of real systems these must either be obtained from experiments or be calculated. In many cases it has proved to be very hard to get all parameters from experiments. In particular lateral interactions are difficult to derive from experimental data. Here another computational advance has shown itself to be very important. Lateral interaction, and also other kinetic parameters, may be obtained from quantum chemical calculations. In particular Density-Functional Theory (DFT) with the Generalized Gradient Approximation (GGA) has been shown to give quite accurate results.[6]

In this paper we show results for the reduction of NO ($2NO \rightarrow N_2 + O_2$) on a Rh(111) surface using DMC simulations with kinetic parameters which are to a large extent calculated with DFT-GGA. This system is important for exhaust gas catalysis. Not only are lateral interactions present in this system, but also three different adsorption sites are involved. We think that this is currently the most complex system for which *ab-initio* kinetics has been done.

2 Computational Details

In this section we briefly discuss the theory of DMC and DFT, and we present the reaction models that we have used.

2.1 Dynamic Monte Carlo Simulations

Three parts can be distinguished in our DMC method: the model representing the catalyst and the adsorbates, the ME that describes the evolution of the system, and the DMC algorithms to solve the ME. The ME is given by[1,7]

$$\frac{dP_\alpha}{dt} = \sum_\beta \left[W_{\alpha\beta} P_\beta - W_{\beta\alpha} P_\alpha \right], \tag{1}$$

where α and β refer to the configuration of the adlayer, the P's are the probabilities of the configurations, t is real time, and the W's are transition probabilities per unit time. These transition probabilities give the rates with which reactions change the occupations of the sites. They are very similar to macroscopic reaction rate constants and we will use this term in the rest of this paper, although one should remember that they refer to reactions at the atomic scale. $W_{\alpha\beta}$ corresponds to the reaction that changes β into α. A configuration is the assignment of the adsorbates to the sites in the system.

The derivation of the ME shows that the rate constants can be written as

$$W_{\alpha\beta} = \frac{k_B T}{h} \frac{Q^\ddagger}{Q} \exp\left[-\frac{E_{\mathrm{bar}}}{k_B T} \right], \tag{2}$$

with k_B the Boltzmann-constant, h Planck's constant, T temperature, and E_{bar} the activation barrier of the reaction that transforms configuration β into configuration α. The partition function $Q^\ddagger$ and Q can be interpreted as the partition

functions of the transition state and the reactants, respectively, although there are some small, generally negligible, differences.[1] The important point is that this equation makes an *ab-initio* approach to kinetics possible.

The DMC simulations form a powerful numerical method to solve the ME exactly. In fact there are numerous DMC algorithms that might be used: a recent taxonomy of these algorithms contained no less than 48.[4] Most of them are not efficient for any reaction system, however. For a general ME various algorithms have been given by Binder.[8] (Note, however, that here the ME exists before the DMC algorithms, whereas in equilibrium MC it is the other way around. As a consequence t is real time, and not some time in MC steps.) DMC algorithms for rate equations have even been given earlier by Gillespie.[9,10] For lattice-gas systems the algorithms given by Binder and Gillespie can be made much more efficient, and also other algorithms can be used.[2,3,4] All DMC algorithms generate an ordered list of times at which a reaction takes place, and for each time in that list the reaction that occurs at that time. A DMC simulation starts with a chosen initial configuration. The list is traversed and changes are made to the configuration corresponding to the occurring reactions. The various algorithms differ in how the reaction times are computed, how a reaction of a particular type is chosen, and how it is determined where on the surface a reaction takes place.

2.2 Density-Functional Theory Calculations of Kinetic Parameters

DFT calculations have become very popular for doing quantum chemistry, as DFT forms a good combination of efficiency and accuracy.[6] We have done DFT calculations with the VASP code.[11,12] This code solves the Kohn-Sham equations with a plane wave basis set and the ultrasoft pseudopotentials introduced by Vanderbilt and generated by Kresse and Hafner.[13,14] The Generalized Gradient Approximation of Perdew and Wang has been used (DFT-GGA), because it generally yields good bond energies.[15] The calculations were done with an energy cut-off of 400 eV for the basis, k-point sampling of $5 \times 5 \times 1$, and a surface model consisting of a supercell with a slab of five metal layers separated by a vacuum of 13.4 Å. The supercells used have a 2×2, 3×3, or a $c(2 \times 4)$ structure and various combinations of NO, N, and O.[16,17] Adsorbates were put on both sides of the slab so that the system was inversion symmetric, avoiding dipole-dipole interactions between the slabs. The calculations were converged to within $5\,\mathrm{kJ.mol^{-1}}$ for the adsorption energy. The calculations allowed us to calculate the lateral interactions for adsorbates on the following combinations of sites: neighboring top sites, neighboring fcc sites, neighboring hcp sites, an fcc site and the next-nearest top site, an fcc site and the next-nearest hcp site, an hcp site and the next-nearest top site, an hcp site and the next-nearest fcc site, and a top site and the next-nearest fcc and hcp sites (see figure 1). The interactions between adsorbates on a top site and the nearest fcc and hcp sites, and between nearest fcc and hcp sites are very strongly repulsive. These interactions have not been calculated, and we have assumed that adsorbates will never occupy these pairs of sites at the same time.

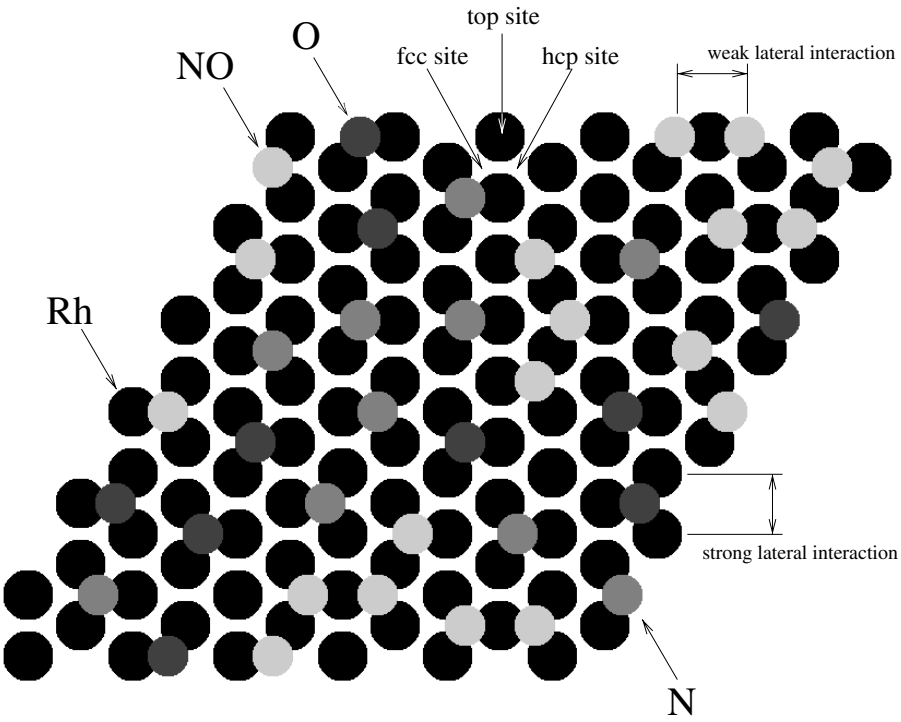


Fig. 1. A snapshot of a small part of the Rh(111) surface with adsorbates during a simulation of the 3-site model. Only the top Rh layer and the adlayer are shown. The double arrows indicate the two distances at which lateral interactions work. Lateral interactions between adsorbates that get closer are extremely repulsive, and the simulations do not allow such close approaches. In the 1-site model only hcp sites (or only fcc sites) are used.

2.3 The Reaction Models

NO, atomic nitrogen, and atomic oxygen all prefer threefold adsorption sites (see figure 1). As a first model, the 1-site model, we have assumed that there is only one threefold site (either the fcc or the hcp) relevant for the kinetics. We have simulated the Temperature-Programmed Desorption (TPD) of NO on Rh(111). In a TPD experiment an adsorbate, in our case NO, is deposited on a catalyst at a low temperature at which no reactions occur. Then the temperature is raised and the rate of desorption is measured. A peak in the desorption rate is generally interpreted as desorption from a particular type of site: the temperature of the peak depends on the bonding energy. Reactions on the surface and lateral interactions can complicate the interpretation of TPD spectra enormously.

The reactions that can occur in our model of the TPD experiment are

$$\mathrm{NO(ads)} + * \ \rightarrow \mathrm{N(ads)} + \mathrm{O(ads)}, \tag{3}$$

$$\mathrm{NO(ads)} \ \rightarrow \mathrm{NO(gas)} + *, \tag{4}$$

$$2\mathrm{N(ads)} \ \rightarrow \mathrm{N_2(gas)} + 2*, \tag{5}$$

where $*$ stands for a vacant site. The two sites involved in all reactions, except for the NO desorption, are nearest neighbors. There is no O_2 desorption in our model because this takes place at much higher temperatures than the other reactions. NO dissociation is suppressed at high coverages, because of site blocking: i.e., it needs a neighboring vacant site, which may not be present. There is an additional suppression due to lateral interaction, which also influence the other reactions. We have included lateral interactions through the Polanyi-Brønsted relation for the activation barrier:[18]

$$E_{\mathrm{bar}} = E_{\mathrm{bar}}^{(0)} + \alpha(\Delta\Phi_{\mathrm{product}} - \Delta\Phi_{\mathrm{reactant}}), \tag{6}$$

were $E_{\mathrm{bar}}^{(0)}$ is the activation barrier without lateral interactions, and $\Delta\Phi_{\mathrm{product}}$ ($\Delta\Phi_{\mathrm{reactant}}$) is the change in adsorption energy of the products (reactants) due to lateral interactions. The coefficient α is the Brønsted-coefficient. It was taken equal to 1 for all reactions (late barrier) except for diffusion for which we used $\alpha = 1/2$. We have assumed that the lateral interactions are pairwise additive: $\Delta\Phi_{\mathrm{product}}$ and $\Delta\Phi_{\mathrm{reactant}}$ are simply the sums of pair interactions. DFT calculations gave no indications that this is incorrect. Only lateral interactions with nearest neighbors were included in the 1-site model. Using realistic diffusion rate constants would make the simulations too costly. The rate constants that we have used (see table 1) are five to eight orders of magnitude smaller than the real ones. However the changes in the results when the rate constants for diffusion are increased are negligible.

In the 3-site model we included both threefold sites (fcc and hcp) and the onefold (top) site in the unit cell. The reason for this more complex model was *not* based on the kinetic results of the simulations with the 1-site model, but on the discrepancy for the lateral interactions between the DFT calculations and fitted values with the 1-site model (see below). Also a LEED study of NO/Rh(111)

Table 1. Activation energies and prefactors for the reactions in the 3-site model when there are no lateral interactions. Nitrogen and oxygen diffuse only between 3-fold sites.

reaction	prefactor (in 1/sec)	activation energy (in kJ/mol)
NO dissociation	10^{11}	65
NO desorption (3-fold)	$10^{13.5}$	99
NO desorption (top)	$10^{13.5}$	52
N_2 desorption	10^{10}	120
NO diffusion (3-fold→3-fold)	10^{5}	22.5
NO diffusion (3-fold→top)	10^{5}	47
NO diffusion (top→3-fold)	10^{5}	0
N diffusion	10^{5}	16
O diffusion	10^{8}	45

with high coverage indicates that NO can be distributed over three sites in our 3-site model.[19] The reactions are the same as for the 1-site model, but, as we have more sites, we have more different lateral interactions.

3 Results and Discussion

The results of the DMC simulations obtained with the 1-site model are in reasonable agreement with the experiment. The kinetic parameters are obtained by fitting to the experimental data. For some parameters this is quite straightforward. For example, the activation energy and the prefactor for N_2 desorption can be obtained from a TPD experiment starting with a low coverage layer of nitrogen that has been atomically deposited.[20] The low coverage is necessary to avoid effects of lateral interactions. Similarly the activation energies and prefactors for NO dissociation can be obtained. It's much harder to obtain kinetic parameters for the NO desorption, because this only occurs when many other adsorbates are present. These adsorbates block the sites necessary for the NO dissociation which would otherwise occur. Their lateral interactions influence, however, the NO desorption. It is also very hard to obtain the lateral interactions themselves from experiments. We have determined these parameters by varying them, by hand, and by fitting the simulated TPD spectra to the experimental ones. There are two problems with the kinetic parameters for the 1-site model. First the TPD experiments do not seem to contain enough data to determine the lateral interactions uniquely. Various sets of them were obtained by different people which gave fits of similar quality. Second, there is a huge discrepancy with DFT results: DFT gives values for the same lateral interactions that are almost an order of magnitude larger than what is obtained from fitting to the TPD spectra. Using the DFT results in the 1-site model gave totally incorrect spectra.

We have extended the 1-site model to the 3-site model to resolve the discrepancy between the fitted and the calculated lateral interactions (see figure 2). The main difference in the structures that the adlayer forms in the simulations

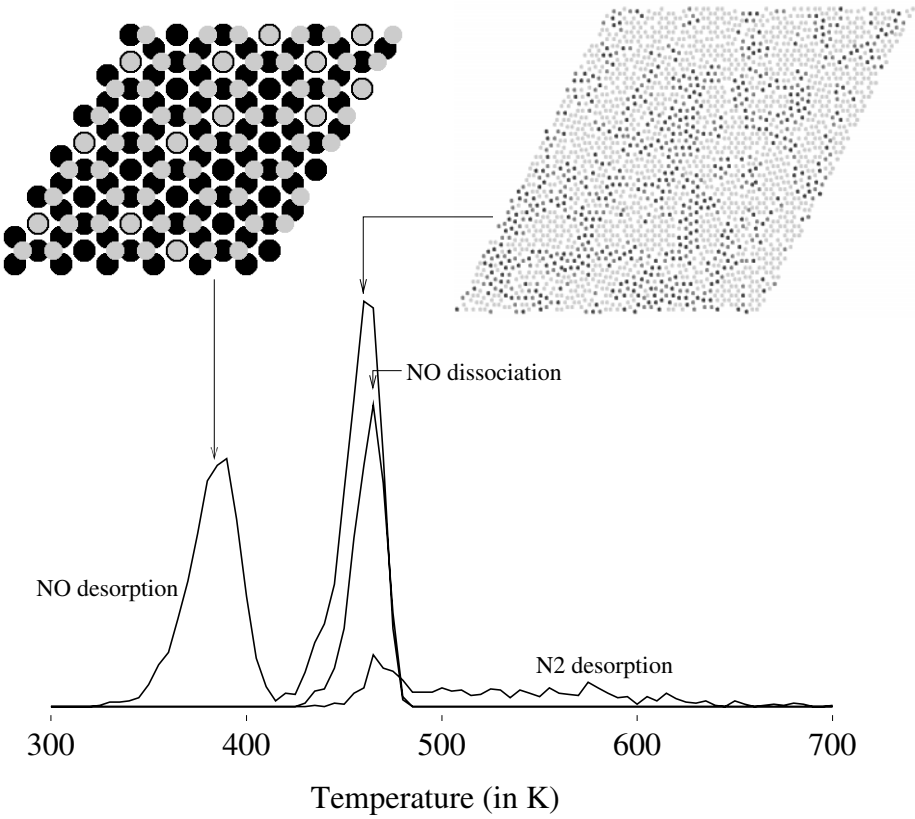


Fig. 2. Snapshots and reaction rates as a function of temperature from a simulation of the TPD experiment with a maximum initial NO coverage ($\theta_{\mathrm{NO}} = 0.75$) and a heating rate of $10\,\mathrm{K.s}^{-1}$. The 3-site model was used. The desorptions can be measured by TPD, but another technique is necessary to measure the NO dissociation. The top-left snapshot shows the top Rh layer and the adsorbates (only NO) of a small part of the system at $T = 385\,\mathrm{K}$. The top-right snapshot shows only the adsorbates of the whole system at $T = 458\,\mathrm{K}$.

with the 1- and the 3-site model is that in the 3-site model the adsorbates can stay farther apart at high coverages. Consequently the system avoids almost completely lateral interactions for which the DFT calculations give high values. Occasionally adsorbates do get as close to each other as in the 1-site model, but only when it's necessary for restructuring the adlayer. These high lateral interactions do not show up in the TPD spectra. The lateral interactions and other kinetic parameters are given in tables 1 and 2.

Table 2. Lateral interactions (in kJ/mol) in the 3-site model for two adsorbates at distances 1 and $2/\sqrt{3}$ times the distance between two Rh atoms. (Interactions between two NO molecules both at top sites have not been calculated because close NO at top sites doe not occur due to the low adsorption energy at top sites.)

	NO (3-fold)		NO (top)		N (3-fold)		O (3-fold)	
NO (3-fold)	26	0	-	5	23.5	16	101	15
NO (top)					-	0	-	6.5
N (3-fold)					40	21.5	45.5	25
O (3-fold)							26	26

The maximum coverage of NO (i.e., the maximum number of NO molecules on the surface per Rh atom in the top layer of the catalyst) is 0.75. The partial occupation of sites of the same type indicates that there are repulsive lateral interactions. In the 3-site model the coverage $\theta_{\mathrm{NO}} = 0.75$ corresponds to the $(2 \times 2) - 3\mathrm{NO}$ structure in which a quarter of each of the top, hcp, and fcc sites are occupied. In this structure the NO molecules feel only the weak repulsive interaction. Putting more NO molecules on the catalyst immediately leads to strong repulsive interactions (see figure 1). In the 1-site model there is only a gradual increase in the lateral interaction energy with coverage, and it's not clear in this model why the maximum coverage should be 0.75.

Dissociation of isolated NO is a fast reaction. As the dissociation products, N and O, need one extra site, the dissociation will not occur, however, when sites are blocked or when lateral interactions increase the activation barrier too much. In that case NO dissociation will only occur after the slower NO desorption has made sufficient room on the catalyst's surface. Figure 2 shows that the NO molecules at the top sites desorb first. They are bound less strongly to the surface. This does not lead to sites useful for NO dissociation yet. Only when NO molecules vacate hcp and fcc sites NO dissociation begins.

At a slightly higher temperature N_2 desorption also starts taking place. The N_2 desorption takes place over a very broad temperature range and the peak at the beginning of that range is also caused by lateral interactions. The difference in lateral interactions between NO, N, and O causes segregation into NO and N + O islands. O_2 desorption only occurs at much higher temperatures and is not shown in figure 2.

4 Conclusions

We have done DMC simulations of the NO reduction to N_2 and O_2 on Rh(111) with kinetic parameters obtained from DFT calculations. Our 3-site model has three different sites per unit cell, site blocking, and lateral interactions for all reactions and for diffusion. The DMC simulations gave results for real time dependence and used exact reaction rate constants for simulating TPD spectra. The DFT calculations were done with a plane wave basis set and ultrasoft pseudo-potentials on a five-layer slab and supercells up to 3×3 structures.

The present study shows the necessity of DMC simulations with DFT calculations to understand kinetics in heterogeneous catalysis. A 1-site model shows that it's well possible to reproduce the experimental kinetics, but that this does not mean that the model is correct. Only the comparison between fitted and calculated kinetic parameters shows the shortcomings of this too simple model. The results of the 3-site model with kinetic parameters obtained from DFT calculations indicates that it is becoming possible to do ab-initio kinetics for real-world applications.

References

1. Gelten, R. J., van Santen, R. A., and Jansen, A. P. J.: Dynamic Monte Carlo simulations of oscillatory heterogeneous catalytic reactions. In P. B. Balbuena and J. M. Seminario, editors, *Molecular Dynamics: From Classical to Quantum Methods*. Elsevier, Amsterdam (1999)
2. Jansen, A. P. J.: Monte Carlo simulations of chemical reactions on a surface with time-dependent reaction-rate constants. Comput. Phys. Comm. **86**, (1995) 1–12
3. Lukkien, J. J., Segers, J. P. L., Hilbers, P. A. J., Gelten, R. J., and Jansen, A. P. J.: Efficient Monte Carlo methods for the simulation of catalytic surface reactions. Phys. Rev. E **58**, (1998) 2598–2610
4. Segers, J. P. L.: *Algorithms for the Simulation of Surface Processes*. Ph.D. thesis, Eindhoven University of Technology (1999)
5. Carlos is a general-purpose program, written in C by J. J. Lukkien, for simulating reactions on surfaces that can be represented by regular grids; an implementation of the first-reaction method, the variable stepsize method, and the random selection method. Write to `tgtatj@chem.tue.nl` if you are interested in using Carlos.
6. Koch, W. and Holthausen, M. C.: *A Chemist's Guide to Density Functional Theory*. Wiley-VCH, New York (2000)
7. van Kampen, N. G.: *Stochastic Processes in Physics and Chemistry*. North-Holland, Amsterdam (1981)
8. Binder, K.: *Monte Carlo Methods in Statistical Physics*. Springer, Berlin (1986)
9. Gillespie, D. T.: A general method for numerically simulating the stochastic time evolution of coupled chemical reactions. J. Comput. Phys. **22**, (1976) 403–434
10. Gillespie, D. T.: Exact stochastic simulations of coupled chemical reactions. J. Phys. Chem. **81**, (1977) 2340–2361
11. Kresse, G. and Furthmüller, J.: Efficiency of ab-initio total energy calculations for metals and semiconductors using a plane-wave basis set. Comput. Mat. Sci. **6**, (1996) 15–50

12. Kresse, G. and Furthmüller, J.: Efficient iterative schemes for ab initio total-energy calculations using a plane-wave basis set. Phys. Rev. B **54**, (1996) 11169–11186
13. Vanderbilt, D.: Soft self-consistent pseudopotentials in a generalized eigenvalue formalism. Phys. Rev. B **41**, (1990) 7892–7895
14. Kresse, G. and Hafner, J.: Norm-conserving and ultrasoft pseudopotentials for first-row and transition elements. J. Phys.: Condens. Matter **6**, (1994) 8245–8257
15. Perdew, J. P.: Unified theory of exchange and correlation beyond the local density approximation. In P. Ziesche and H. Eschrig, editors, *Electronic Structure of Solids '91*, 11. Akademie Verlag, Berlin (1991)
16. Loffreda, D., Simon, D., and Sautet, P.: Molecular and dissociative chemisorption of NO on palladium and rhodium (100) and (111) surfaces: A density-functional periodic study. J. Chem. Phys. **108**, (1998) 6447–6457
17. Thomas, J. M. and Thomas, W. J.: *Principles and Practice of Heterogeneous Catalysis*. VCH, Weinheim (1997)
18. van Santen, R. A. and Niemantsverdriet, J. W.: *Chemical Kinetics and Catalysis*. Plenum Press, New York (1995)
19. Zasada, I., Hove, M. A. V., and Somorjai, G. A.: Reanalysis of the Rh(111) + $(2 \times 2) - 3$NO structure using automated tensor leed. Surf. Sci. **418**, (1998) L89–L93
20. van Hardeveld, R. M., van Santen, R. A., and Niemantsverdriet, J. W.: Formation of NH_3 and N_2 from atomic nitrogen, and hydrogen on rhodium(111). J. Vac. Sci. Technol. A **15**, (1997) 1558–1562

Interpolating Wavelets in Kohn-Sham Electronic Structure Calculations

A.J. Markvoort, R. Pino, and P.A.J. Hilbers

Technische Universiteit Eindhoven,
Department of Computing Science,
Postbus 513, 5600 MB Eindhoven,
The Netherlands.
A.J.Markvoort@tue.nl

Abstract. In many biology, chemistry and physics applications quantum mechanics is used to study material and process properties. The methods applied are however expensive in terms of computational as well as memory requirements and scale poorly. In this work we describe an alternative method based on wavelets with better scaling properties. We show how the Kohn-Sham equations, both spin polarized and spin unpolarized, are solved and give a description of pseudopotentials and a preconditioned conjugate gradient method to solve the Hartree potential and the Schrödinger equation. Example calculations for small molecules are given to show the validity of the method.

1 Introduction

Most of low-energy physics, chemistry and biology can be explained by the quantum mechanics of electrons and ions. First-principles methods based on density functional theory have proven to be an accurate and reliable tool in understanding and predicting a wide variety of physical and chemical properties [1]. Traditional ab-initio methods are however extremely expensive in terms of computational as well as memory requirements. Typically, the computer time scales as N^3 where N is the number of electrons in the system, restraining the system sizes that can be examined. In order to treat grand challenge problems, such as computational description of catalytic processes, a significant increase in computational power is required or new methods have to be devised with better scaling properties. One such method based on wavelets is described in this paper.

In section 2 wavelets are discussed. In section 3 the method used to solve the Kohn-Sham equations is described. In section 4 numerical results of some example calculations are given and we finish with some conclusions.

2 Interpolating Wavelets

Most methods for solving electronic structure calculations employ plane waves or atomic orbitals (LCAO). Plane waves have the advantage of being orthonormal

V.N. Alexandrov et al. (Eds.): ICCS 2001, LNCS 2073, pp. 541–550, 2001.
© Springer-Verlag Berlin Heidelberg 2001

and complete, permitting systematic convergence and straightforward evaluation of forces, which is essential for the extension to molecular dynamics simulations. But they are not efficient in describing localized orbitals and wavefunctions in surfaces or clusters.

On the other hand localized bases, like LCAO or Gaussians, are usually over-complete, lack explicit convergence properties and result in difficult force calculations.

However, the well known fact that electronic wave functions vary much more rapidly near the atomic nuclei than in inter-atomic regions calls for a multiresolution approach. This is provided by an alternative basis, formed of wavelets [2]. This alternative was first presented by Cho *et al.* [4] who employed (Mexican hat) wavelets in solving the Schrödinger equation for Hydrogen like atoms. The idea is that a wavelet basis set combines the desirable properties of both localized as well as plane wave basis sets. Its main advantage lies in its capability to provide a multiresolution analysis, allowing one to use low resolution and to add extra resolution only in those regions where necessary. In this way, wavelets as basis sets allow accurate description over a range of length scales.

Later, self-consistent LDA calculations on H_2 and O_2 using Daubechies wavelets were reported [5,6]. The use of interpolating wavelets has been introduced recently by Lippert *et al.* [7]. A review on wavelets in electronic structure has been given by Arias [3].

The multiresolution property of the wavelets is related to the dilation equation

$$\phi(x/2) = \sum_{l \in Z} h_l \phi(x - l). \tag{1}$$

Figure 1 shows how a wavelet at one level is, according to this dilation equation, the weighted sum of wavelets at one level lower.

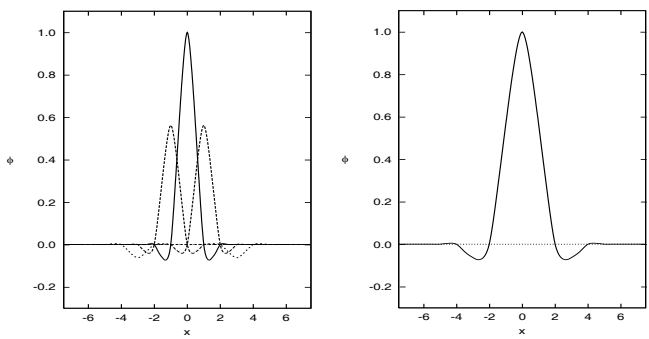


Fig. 1. Construction of an interpolet using the dyadic equation. An interpolet (right) is the weighted sum of interpolets at one level lower (left).

Because of the freedom in choosing the filter coefficients h_l, very many different kinds of wavelets exist. This freedom can be used to give the wavelets other

favorable properties. The interpolating wavelets, interpolets for short, were constructed by Donoho [9]. The reason we have chosen for these wavelets for our basis is that they combine the multiresolution property of general wavelets with the smoothness of interpolation.

Within the family of interpolets there are again various members. These differ in the number of non-zero filter coefficients h_l. This number of non-zero filter coefficients is related to the support length of the interpolet, i.e. the range in which the interpolet is non-zero. The number of non-zero filter coefficients also determines the degrees of freedom to imply the other properties of the interpolets. First, cardinality is implied, i.e. $\phi(k) = \delta_{k,0},\ \forall k \in Z$. The remaining freedom is used for polynomial span. This means that the higher the number of non-zero filter coefficients, the higher the degree M of polynomials that can be represented exactly.

The basis we employ consists of interpolets of different resolution levels j and of different centers k, i.e. various dilations and translations of one mother function. A function f is thus expanded as

$$f(x) = \sum_j \sum_k s_k^j \phi^j(x-k). \tag{2}$$

An important aspect is that not all possible translations and dilations have to be present. If high frequency oscillations are present only in a small range, only in this range narrow interpolets are needed, whereas it suffices to use only broader interpolets in the remaining space, resulting in a truncated basis. Each interpolet coefficient s_k^j can be calculated from the functional values of f in a fixed number of calculations. The interpolet transform is thus linear in the number of interpolets used.

In order for the basis to be useful we want to perform operator actions on the interpolets. The action on the interpolets should be written in terms of the interpolets themselves. An example of an operator needed in our calculations is the second derivative operator. For interpolets in the same resolution level a relation

$$L_l^0 = \int dx\ \phi(x-l)\frac{d^2}{dx^2}\phi(x) \tag{3}$$

is obtained. By substituting the dilation equation one obtains the recursive relation [10]

$$L_l^0 = 2\sum_{l_1}\sum_{l_2} h_{l_1} h_{l_2} L_{2l+l_1-l_2}. \tag{4}$$

Because of the compact support of the interpolets, most coefficients L_l^0 are equal to zero and the non-zero elements can be calculated. From these the inter-level coefficients can be calculated as well, again, using the dilation equation. Using these coefficients the action of the operator can be calculated in linear time with the number of basis functions used.

Interpolets in three dimensional space are created as a tensor product of one dimensional versions.

3 Approach

A method to resolve the electronic structure is by using a variational principle:

$$E(\Psi) = \frac{\langle \Psi | \hat{H} | \Psi \rangle}{\langle \Psi | \Psi \rangle}, \tag{5}$$

where $\langle \Psi | \hat{H} | \Psi \rangle = \int d\mathbf{r}\, \Psi^*(\mathbf{r}) \hat{H} \Psi(\mathbf{r})$, Ψ denotes the electronic wave function and $\hat{H}$ the Hamiltonian. The energy computed from a guess Ψ is an upper bound to the true ground state energy E_0. Full minimization of the functional $E(\Psi)$ will give the true ground state Ψ^{gs} and energy $E_0 = E(\Psi^{gs})$.

Density functional theory states that the many electron problem can be replaced by an equivalent set of self-consistent one-electron equations, the Kohn-Sham equations

$$\hat{H}\Psi_i^\sigma(\mathbf{r}) = \left(-\frac{1}{2}\nabla^2 + \hat{V}_{pp}(\mathbf{r}) + \hat{V}_H(\mathbf{r}) + \hat{V}_{xc}^\sigma(\mathbf{r}) \right) \Psi_i^\sigma(\mathbf{r}) = \epsilon_i^\sigma \Psi_i^\sigma(\mathbf{r}). \tag{6}$$

The eigenfunctions Ψ_i^σ are the one-electron wavefunctions that correspond to the minimum of the Kohn-Sham energy functional. In these wavefunctions, i is the orbital index and σ denotes the spin, which can be either up $\uparrow$ or down $\downarrow$.

The Hamiltonian $\hat{H}$ consists of four different parts: a part related to the kinetic energy of the electrons, the pseudopotential $\hat{V}_{pp}$, the Hartree potential $\hat{V}_H$ and the exchange correlation potential $\hat{V}_{xc}$.

The interaction of the positively charged nuclei with the electrons is described using the pseudopotential $\hat{V}_{pp}$ instead of using the full Coulombic potential. The pseudopotential usually consists of both a local and a non-local part

$$\hat{V}_{pp}(\mathbf{r}) = V_{local}(r) + \sum_l |l\rangle \hat{V}_l(r, r') \langle l|. \tag{7}$$

The Hartree potential $\hat{V}_H$ describes the interaction between electrons and is given by

$$\hat{V}_H(\mathbf{r}) = \int d\mathbf{r}' \frac{\rho_\uparrow(\mathbf{r}') + \rho_\downarrow(\mathbf{r}')}{|\mathbf{r} - \mathbf{r}'|}. \tag{8}$$

Finally, the exchange correlation potential $\hat{V}_{xc}$ describes the non classical interaction between the electrons and is given by the functional derivative of an exchange correlation energy functional

$$V_{xc}^\sigma(\mathbf{r}) = \frac{\delta E_{xc}(\rho_\uparrow, \rho_\downarrow)}{\delta \rho_\sigma}. \tag{9}$$

In these equations ρ^σ is the electron spin density, defined as

$$\rho_\sigma(\mathbf{r}) = \sum_i f_i^\sigma |\Psi_i^\sigma(\mathbf{r})|^2, \tag{10}$$

where f_i^σ is the occupation number, i.e. the number of electrons in orbital i. In case of LSD every orbital can contain at most one electron. In case of LDA where there is no longer a distinction between spin up and spin down, orbitals can contain at most two electrons.

As can be seen from eqs. (6) to (9) the Hamiltonian $\hat{H}$ depends on the density and via eq. (10) thus on the wavefunctions. This system of non-linear coupled differential equations can be solved self-consistently. Figure 2 gives a schematic overview of the approach used. We start with an initial guess for the orbital wavefunctions $\{\Psi_i^0\}$. The corresponding electron density is then calculated using eq. (10). Given this density ρ the Hartree potential, the exchange correlation potential and the non-local part of the pseudopotential are calculated.

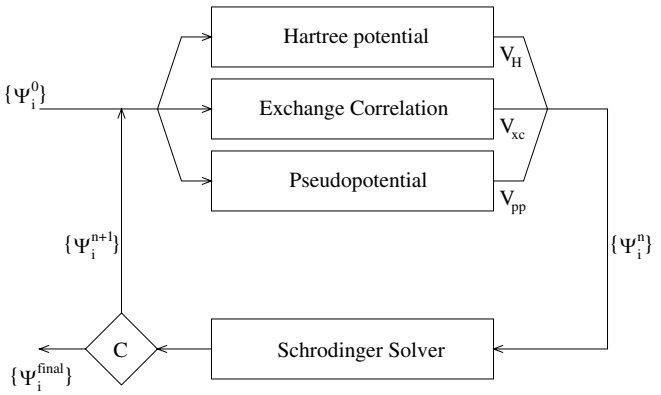


Fig. 2. Scheme used for solving the Kohn Sham equation.

Once the potentials have been calculated, they are kept constant. As a result the Hamiltonian does no longer depend on the wavefunction Ψ and we can use a steepest descent or conjugate gradient method to solve the remaining minimization problem.

After some iterations of this minimization the wavefunction and thus the density will be changed so much that the potentials have to be updated to this new wavefunction. The number of steps before the potentials are recalculated can be chosen fixed or it can depend on a convergence criterion. Given the new potentials the energy minimization is started again, etc. This procedure is repeated till self-consistency.

The rest of this section describes the various blocks in the scheme in more detail.

3.1 Hartree Potential

An important part of the electronic structure calculations is to solve the type of integrals as in eq. (8). However, such integrations are very costly, and problematic

because of the singularity. Instead of calculating the Hartree potential in this way directly, it can also be calculated by solving the Poisson equation

$$\nabla^2 V_H(\mathbf{r}) = -4\pi\rho(\mathbf{r}). \tag{11}$$

Written in terms of wavelets, this is equivalent to solving the set of linear equations $\mathbf{Ls} = \mathbf{r}$, where $\mathbf{L}$ is the matrix that represents the laplacian in wavelet space, $\mathbf{s}$ are the wavelet coefficients of the potential and $\mathbf{r}$ are the wavelet coefficients of the density.

Solving straightforwardly $\mathbf{s} = \mathbf{L}^{-1}\mathbf{r}$ is not the way to go. Namely, $\mathbf{L}$ is singular so the inverse does not exist. And even if it would exist, it would probably, contrary to the original matrix $\mathbf{L}$, not be a sparse matrix. This results in an inefficient calculation, quadratic in the number of basis functions. A better way is to use an iterative procedure like the conjugate gradient method to minimize the function $f(\mathbf{s}) = \frac{1}{2}\mathbf{sLs} - \mathbf{rs}$, since in this minimum $\nabla f(\mathbf{s}) = \mathbf{Ls} - \mathbf{r} = \mathbf{0}$. Because of the large condition number of the matrix representing the laplacian a preconditioner is used to improve the convergence.

For periodized wavelets the matrix L is singular because of the existence of multiple solutions. This can be resolved by applying boundary conditions through a constraint, e.g using a penalty function.

3.2 Pseudopotential

It is well known that most physical and chemical properties are dependent on the valence electrons to a much greater degree than that of the tightly bound core electrons. It is for this reason that pseudopotentials can be used. This has the advantage of a much smaller number of electrons and smoother orbitals, where the singularity in the normal electron-nucleus interaction is removed.

Various pseudopotentials are described in the literature. We have chosen to implement both simple versions and state of the art ones. The Shaw [12] pseudopotential and the Topp-Hopfield [13] pseudopotential are examples of simple, completely local pseudopotentials. The Bachelet-Hamann-Schluter [14] pseudopotential is more advanced but especially suited for plane waves what makes the non local part hard to implement. The Hartwigsen-Goedecker-Hutter [15] pseudopotential is state of the art and also very well suited for grid methods.

The local part of the pseudopotentials only depends on the positions of the nuclei. Thus for a fixed nuclear configuration this only has to be calculated once. The non-local part however depends on the wavefunction and thus has to be recalculated in every step.

3.3 Exchange Correlation Potential

The first approximation for the exchange correlation potential is the local spin density approximation LSD.

$$E_x^{LSD} = -\int d\mathbf{r}\; c_x \left(\rho_\uparrow(\mathbf{r})^{4/3} + \rho_\downarrow(\mathbf{r})^{4/3}\right). \tag{12}$$

The corresponding exchange potentials are

$$V_x^\sigma(\mathbf{r}) = -\frac{4}{3} c_x \rho_\sigma(\mathbf{r})^{1/3}. \tag{13}$$

where $c_x = \frac{3}{4}\left(\frac{6}{\pi}\right)^{1/3}$. In case of LDA this reduces to $V_{xc}^{LDA}(\mathbf{r}) = -\frac{4}{3} c_x \rho(\mathbf{r})^{1/3}$ where the constant c_x is equal to $\frac{3}{4}\left(\frac{3}{\pi}\right)^{1/3}$. More precise approximations for the exchange correlation potential do not only use the density, but also the gradient of the density $\nabla\rho$. These methods are called generalized gradient approximations. We implemented one by Becke [16] and one by Perdew, Burke and Ernzerhof [17]. In these references only the energy functionals are given. The corresponding potentials can be derived by taking the functional derivative to the density. These potentials will not only depend on the density and its gradient, but also on the laplacian of the density.

3.4 Schrödinger Equation

The way we look for the ground state of the electronic wavefunction is by minimizing the total energy given by eq. (5). This total energy is related to the energies E_i of the individual orbitals Ψ_i, i.e.

$$E_i(\Psi_i) = \frac{\langle \Psi_i | \bar{H} | \Psi_i \rangle}{\langle \Psi_i | \Psi_i \rangle}, \tag{14}$$

where the orbital wavefunctions Ψ_i satisfy the appropriate orthonormalization conditions.

The minimization procedure we use is a steepest descent or conjugate gradient method. In such a method the wavefunction is updated in a certain direction g_i

$$\Psi_i^{(n+1)} = \Psi_i^{(n)} + \lambda_i^{(n)} g_i^{(n)}. \tag{15}$$

In the steepest descent method, this direction is the direction of the gradient the wavefunction, which can be calculated by [11]

$$\frac{\delta E_i(\Psi_i)}{\delta \Psi_i^*} = \bar{H}\Psi_i. \tag{16}$$

Minimizing in the direction of the gradient seems a natural way to work. However, it has been proven that it is much more efficient to regard also the search directions of previous steps. This is employed in the conjugate gradient method.

The only free parameter we have is λ. This determines the step size and should be chosen such that the energy is minimized in our search direction. A simple method is to take a fixed (small) step size. However, this does probably not bring us to the minimum in our search direction. An optimal value for $\lambda_i^{(n)}$ can be derived by substituting equation 15 in eq. (14) and rewriting it as

$$E_i^{(n+1)} = \frac{d + e\lambda + f\lambda^2}{1 + b\lambda + c\lambda^2}, \tag{17}$$

where the coefficients b, c, e and f are integrals that have to be evaluated and d is the old energy $E_i^{(n)}$. Given these coefficients the minimum of the function for the energy, eq. (17), can be solved analytically.

However, this method only works as long as the Hamiltonian is constant. In general this is not the case. The non-local part of the pseudopotential, the Hartree potential and the exchange and correlation potential depend on the wavefunction and thus change together with this wavefunction in every step of the iteration. Because of this, the energy is minimized to fixed potentials, which are then updated, giving rise to a new minimization. And this procedure is repeated till self-consistency is reached.

4 Example Calculations

A code has been developed that implements the scheme described above, where interpolating wavelets are used as a basis. Because of the use of pseudopotentials there is an offset in the energies calculated. However, most interesting properties depend on differences of energies, where these offsets cancel each other.

For instance, for single atoms, using differences of energies, ionization and excitation energies can be calculated. Ionization energies for the first 11 elements of the periodic table are given in table 1. All calculations have been performed on a grid of 64^3 points using Hartwigsen pseudopotentials, LDA or LSD exchange and no correlation energy.

Table 1. First and second ionization energies for various atoms. Experimental results are from www.webelements.com.

Atom	First ionization energy (eV)		Second ionization energy (eV)	
	calculated	experiment	calculated	experiment
H	13.0	13.6		
He	24.3	24.6	52.8	54.4
Li	5.4	5.4		
Be	8.9	9.3	18.0	18.2
B	9.0	8.3	24.4	25.2
C	12.3	11.3	25.3	24.4
N	15.6	14.5	30.7	29.6
O	14.2	13.6	36.4	35.1
F	17.1	17.4	34.2	35.0
Ne	21.6	21.6	40.6	41.0
Na	5.2	5.1		

For molecules, bond properties can be calculated. The energies of different diatomic molecules have been calculated for different inter-nuclei distances (R). As an example, the results for carbon monoxide CO has been plotted in figure 3.

The calculated points around the minimum are fitted using a parabola because close to the bond length the bond can be assumed to be elastic. Some bond

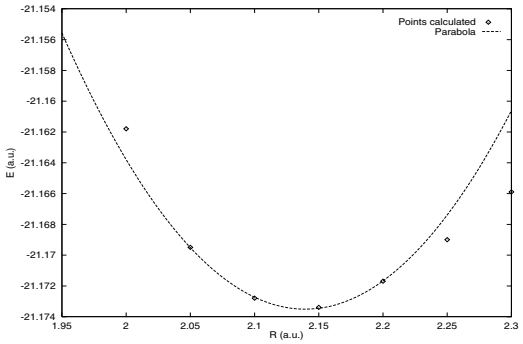


Fig. 3. Energy vs. inter-nuclei distance for carbon monoxide (CO).

properties can be derived from the combination of such a figure and the energy of the separate atoms. In the first place the bond length can be found as the inter-nuclei distance (R) for which the energy of the molecule E_{AB} is minimal. Secondly, the binding energy can be determined by the difference of the energy of the molecule minus the energies of the two separate atoms $E_{bond} = E_{AB} - E_A - E_B$. Thirdly, the bond strength, which can be described both with the bond frequency or with the force constant. The resulting properties for carbon monoxide and for the hydrogen molecule H_2 are given in table 2.

Table 2. Comparison of bond properties as calculated with experimental results[18] for the hydrogen molecule (H_2) and for carbon monoxide (CO).

	H_2			CO		
	LDA	LSD	Exp.	LDA	LSD	Exp.
Binding energy (eV)	6.30	4.90	4.52	16.80	10.76	11.16
Bond length $(a.u.)$	1.48	1.44	1.40	2.14	2.12	2.13
Force constant (N/cm)	4.48	5.41	5.75	15.57	17.98	19.02
Bond frequency (cm^{-1})	3882	4269	4401	1963	2109	2169

The results in tables 1 and 2 show good agreement with results reported in literature obtained using Kohn-Sham calculations with different basis sets.

5 Concluding Remarks

The Kohn-Sham equations, both spin polarized and spin unpolarized, were solved using a basis consisting of interpolating wavelets. The validity of the scheme was demonstrated using some example calculations.

So far only electronic structure has been considered, but the intention is to extend the code to an ab-initio molecular dynamics code. An important part here is the pruning, i.e. the selection of the interpolets that can be left out, resulting in an as small as possible basis. Another improvement would be the development of special pseudopotentials for wavelets.

References

1. For a recent review see e.g. Comp. Phys. Comm. **128** (2000) 1-530
2. See e.g. Daubechies, I.: Ten Lectures on Wavelets. SIAM (1992)
3. Arias, T.A.: Multiresolution Analysis of Electronic Structure - Semicardinal and Wavelet Bases. Rev. Mod. Phys. **71** (1999) 267-311
4. Cho, K., Arias, T.A., Joannopoulos, J.D., Lam, P.K. Wavelets in Electronic Structure Calculations Phys. Rev. Lett. **71** (1993) 1808-1811
5. Wei, S., Chou, M.Y.: Wavelets in Self-Consistent Electronic Structure Calculations Phys. Rev. Lett. **76** (1996) 2650-2653
6. Tymczak, C.J., Wang, X. Orthonormal Wavelet Bases for Quantum Molecular Dynamics Phys. Rev. Lett. **78** (1997) 3654-3657
7. Lippert, R.A., Arias, T., Edelman, A.: Multiscale computations with interpolating scaling functions. J. Comp. Phys. **140** (1998) 278-310
8. Deslaurier, G., Dubuc, S.: Symmetric Iterative Interpolation Process. Constr. Approx. **5** (1989) 49-68
9. Donoho, D.L.: Interpolating Wavelet Transforms. Preprint, Department of Statistics, Stanford university (1992)
10. Beylkin, G.: On the representation of operators in bases of compactly supported wavelets. SIAM J. Numer. Anal. **6** (1992) 1716-1740
11. Stich, I., Car, R., Parrinello, M., Baroni, S.: Conjugate gradient minimization of the energy functional: A new method for electronic structure calculation. Phys. Rev. B **39** (1989) 4997-5004
12. Shaw, R.W.: Optimum form of a modified Heine-Abarenkov model potential for the theory of simple metals. Phys. Rev. **174** (1968) 769-781
13. Topp, W.C., Hopfield, J.J.: Chemically motivated pseudopotential for sodium. Phys. Rev. B **7** (1973) 1295-1303
14. Bachelet, G.B., Hamann, D.R., Schlüter, M.: Pseudopotentials that work. Phys. Rev. B **26** (1982) 4199–4228
15. Hartwigsen, C., Goedecker, S., Hutter, J.: Relativistic separable dual-space Gaussian pseudopotentials from H to Rn. Phys. Rev. B **58** (1998) 3641-3662
16. Becke, A.D.: Density-functional exchange-energy approximation with correct asymptotic behavior. Phys. Rev. A **38** (1988) 3098-3100
17. Perdew, J.P., Burke, K., Ernzerhof, M.: Generalized gradient approximation made simple. Phys. Rev. Lett. **77** (1996) 3865-3868
18. Lide, D.R., Frederikse, H.P.R.: CRC Handbook of Chemistry and Physics. CRC Press 1993

Simulations of Surfactant-Enhanced Spreading

Sean McNamara[1,3], Joel Koplik[1], and Jayanth R. Banavar[2]

[1] Benjamin Levich Institute,
City College of New York,
New York, NY 10031
[2] Department of Physics and Center for Materials Physics,
Pennsylvania State University,
University Park, PA 16802
[3] Centre Eurpéen de Calcul Atomique et Moléculaire
Ecole Normale Supériuere de Lyon
46, allée d'Italie
69364 Lyon Cedex 07, France

Abstract. We use computer simulation to study the effect of surfactants on a drop spreading on a solid surface. Surfactants enhance spreading, especially when the hydrophobic head of the surfactant molecule is strongly attracted to the solid. A significant part of the spreading is due to the "shielding" of the surface by the surfactant. We use a novel boundary condition that reduces the simulation time by a factor of two.

1 Introduction

It is an experimental fact that certain surfactants facilitate the spreading of drops on solid surfaces. For certain combinations of surfactants, the increase in spreading is truly significant [1]. We investigate this phenomena using molecular dynamics simulations.

Surfactants could enhance spreading through two mechanisms [2]. First of all, surfactants reduce the surface tension, causing the drop to spread out. A second mechanism is possible if, for example, a drop of water is placed on a greasy surface. A surfactant, such as soap, with a hydrophobic and and a hydrophilic end could interpose itself between the water and the surface, thus shielding the water from the surface. The drop would the spread more easily for the repulsive interactions between the fluid and the surface would be reduced. In this paper, we emphasize the second "shielding" mechanism and show that it is important.

In this paper, we perform several computer "experiments" where a drop is placed on a solid surface, and allowed to spread. We first examine the case without surfactants. The final state of the drop depends delicately on the interaction between the liquid and the solid. We then fix the liquid-solid interaction and experiment with different surfactants, looking for the surfactants which enhance the spreading the most. There are many different parameters, so this paper does not come close to exhausting the possibilities. We investigate three different

V.N. Alexandrov et al. (Eds.): ICCS 2001, LNCS 2073, pp. 551–559, 2001.
© Springer-Verlag Berlin Heidelberg 2001

parameters: the relative sizes of the hydrophobic and hydrophilic parts, the solubility of the surfactant and the surfactant-solid interaction. This last parameter is the most significant.

2 The Experiment

2.1 Materials

In our experiments, a drop containing surfactants is placed on a solid surface. Our implementation of surfactants [3] and spreading [4] follows previous work. We define four different types of atoms named A, B, C, and D. All the necessary components are made of these molecules. A sketch of these components is shown in Fig. 1. The fluid is composed of A_2: dimers of atoms of type A. We use dimers instead of monomers because dimers are less volatile. The surfactant has the general form $B_m C_n$. It must be composed of two types of atoms because it has a hydrophilic end and a hydrophobic end. The hydrophilic end is composed of atoms of type B and the hydrophobic end is made of atoms of type C. We will use the terms "hydrophobic" and "hydrophilic" to refer to the two ends of the polymers, even though we do not attempt to make the liquid A_2 mimic water. (Perhaps more appropriate names would be "A-philic" and "A-phobic".) Finally, the solid is composed of atoms of type D.

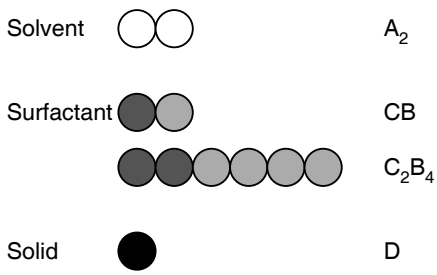


Fig. 1. The different molecules used in the simulations.

The potential between all the atoms is built out of the Lennard-Jones potential [5]:

$$V_{LJ}(r) = 4\epsilon \left[\left(\frac{\sigma}{r} \right)^{-12} - C_{\alpha\beta} \left(\frac{\sigma}{r} \right)^{-6} \right]. \quad (1)$$

Here ϵ is the unit of energy, and σ is the effective diameter of the atoms. The potential consists of two parts: a strong short-range repulsion (proportional to r^{-12}), and a longer range attractive potential (proportional to r^{-6}). We have added the factor $C_{\alpha\beta}$ which depends on the species of the interacting atoms. By changing $C_{\alpha\beta}$, one can increase or decrease the attractive force between the

molecules. In this paper, $C_{AA} = 1$, so V_{LJ} reduces to the traditional Lennard-Jones potential. We create hydrophilic and hydrophobic materials by setting $C_{AB} > 1$ and $C_{AC} < 1$. C_{AD} controls the behavior of the liquid on the surface. Roughly speaking, $C_{AD} < 1$ gives a liquid which is partially wetting, and $C_{AD} > 1$ gives a wetting liquid. When C_{AD} is much larger than 1, terraced spreading is observed.[4]

Unless otherwise specified, the interaction coefficients $C_{\alpha\beta}$ have the following values: $C_{AB} = 2$ and $C_{AC} = 0$, so that the hydrophilic end is very hydrophilic indeed; it attracts fluid atoms more strongly than the fluid atoms attract one another. But between the hydrophobic end and the fluid atoms, there is no long range (r^{-6}) force, only a short range repulsion. Within the surfact, we set $C_{BB} = 1$ and $C_{BC} = C_{CC} = 0$. Between the solid and the surfactant we have $C_{BD} = 0$ and $C_{CD} = 1$. Thus the hydrophobic end of the surfactant is attracted to the solid.

The interaction potential in (1) extends to infinity, which means that every particle directly influences every other particle. In order to solve for the motion of N particles, we would have to calculate $N^2/2$ interactions, which is too expensive. Therefore, we will "cut" the potential at a cutoff radius r_c, and set $U = 0$ for $r > r_c$. In this paper, we choose $r_c = 2.5\sigma$. We also "shift" the potential so that energy and force are continuous at $r = r_c$. The potential becomes

$$V(r) = \begin{cases} V_{LJ}(r) - V'_{LJ}(r_c)r - V_{LJ}(r_c), & r < r_c \\ 0, & r \geq r_c. \end{cases} \tag{2}$$

Note that "cutting" and "shifting" the potential changes the properties of the fluid. One must take care when comparing results, because some people shift the potential in different ways, or do not shift it at all. Our method of shifting the potential minimizes integration errors.

Neighboring atoms in molecules do not interact via (2), rather they are bonded together by

$$V_{\text{bond}}(r) = 4\epsilon \left[\left(\frac{\sigma}{r}\right)^{-12} + \left(\frac{\sigma}{r}\right)^{6} \right]. \tag{3}$$

Note that the -6 power has been replaced with a $+6$. This means that infinite energy is required to break bonds. The -12 power again keeps the atoms separated.

The solid is made by anchoring atoms of type D to a regular array of lattice sites. The D atoms feel a force $\mathbf{F}$ given by

$$\mathbf{F} = -\left(\frac{\epsilon}{\sigma^2}\right) K(\mathbf{r} - \mathbf{r}_0), \tag{4}$$

where $\mathbf{r}$ is the location of the atom and $\mathbf{r}_0$ is the location to which it is tethered. The spring constant K is set to 50 in this paper. The mass of the D atoms is set to 50 so that its resonant frequency will be of order 1. The other atoms have mass 1.

2.2 Geometry

As shown in Fig. 2, we will consider the spreading of a cylindrical drops on a flat surface. The surface is a plane perpendicular to the z axis, and the drops axis is parallel to the y axis. The experiment is macroscopically uniform along the y axis. The boundary conditions match this assumption by imposing periodicity in the y direction. In all other directions, a force field prevents particles from escaping from the box. In our experiments, we choose $L_x = 140\sigma$, $L_y = 12\sigma$ and $L_z = 60\sigma$.

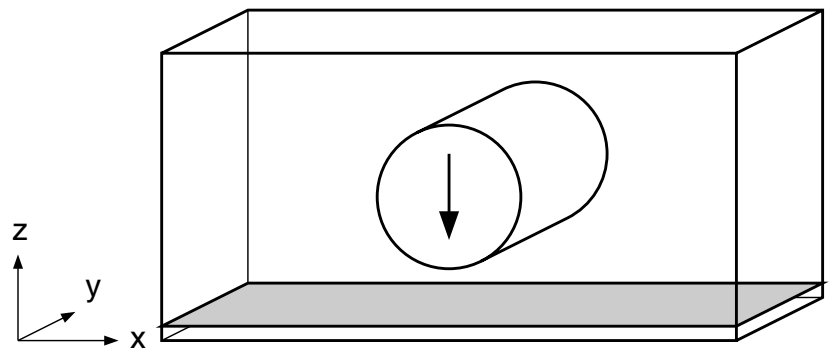


Fig. 2. A sketch of the spreading experiments. Periodic boundary conditions are imposed in the y direction; in all other directions a force field prevents particles from escaping.

In Fig. 3, we show the result of two spreading experiments: one where $C_{AD} = 0.8$ and another where $C_{AD} = 1$. A small change in C_{AD} gives a big change in spreading behavior. When we study surfactants, we will add them to the $C_{AD} = 0.8$ fluid, to try to get it to spread like the $C_{AD} = 1$ fluid.

Note that the simulations of Fig. 3 have approximate mirror symmetry. We would like to exploit this symmetry to simulate only one half of the spreading drop. To do this, we cut the simulation in half at the midplane, and simulate only the left half, as shown in Fig. 4a. At the midplane, we impose a special kind of mirror image boundary conditions. It turns out that the simplest possible mirror image boundary conditions, where a particle at (x, y, z) has an image at $(L_x - x, y, z)$ does not work. The reason is that when a particle approaches the mirror, it always sees its own image approaching from the other side, and it is repelled by that image. This interaction prevents a half-drop from adhering to its image. What is needed is the "shifted mirror" boundary conditions shown in Fig. 4b, where a particle's image(s) are shifted by $L_y/2$. In this way, a particle never interacts directly with its own image, and a cluster of particles placed near the mirror boundary spontaneously forms a half drop.

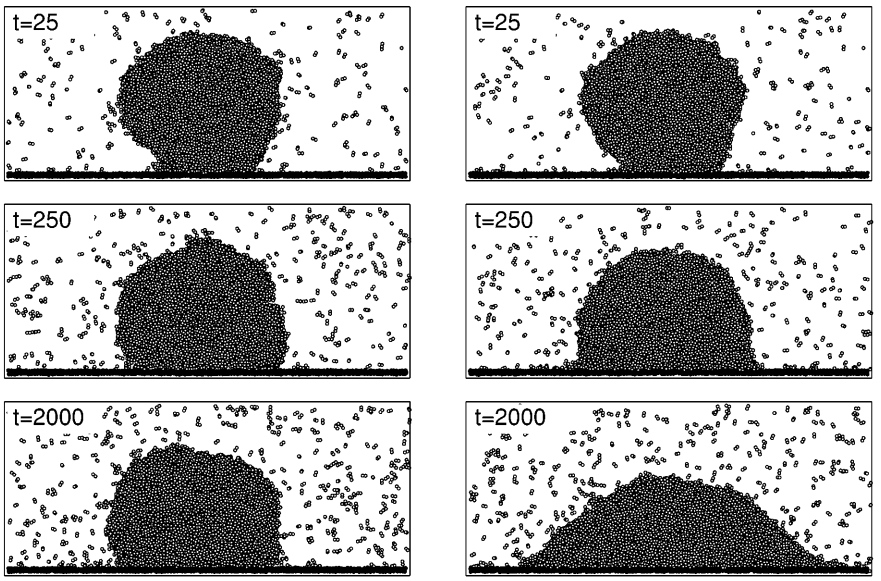


Fig. 3. Results of spreading experiments, with pure fluid: $C_{AD} = 0.8$ (left) and $C_{AD} = 1$ (right). These simulations involve 9000 molecules of A_2 and 1656 solid atoms.

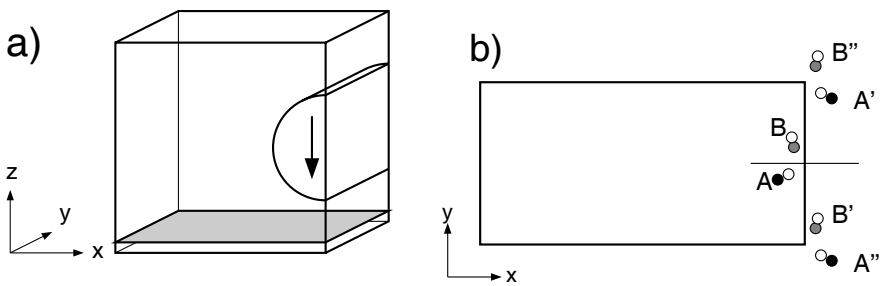


Fig. 4. Boundary conditions for half-drop spreading experiments: a) a sketch of the half-drop experiment; b) the "shifted mirror" boundary conditions applied at the right hand wall (note that the z axis points out of the page in this panel).

In Fig. 5, we compare the results of simulations shown in Fig. 3 with two equivalent simulations using the shifted mirror boundary conditions. The results show that, within fluctuations, the two boundary conditions are equivalent. All the rest of the simulations presented in this paper use shifted mirror boundary conditions, since they run twice as fast as full simulations.

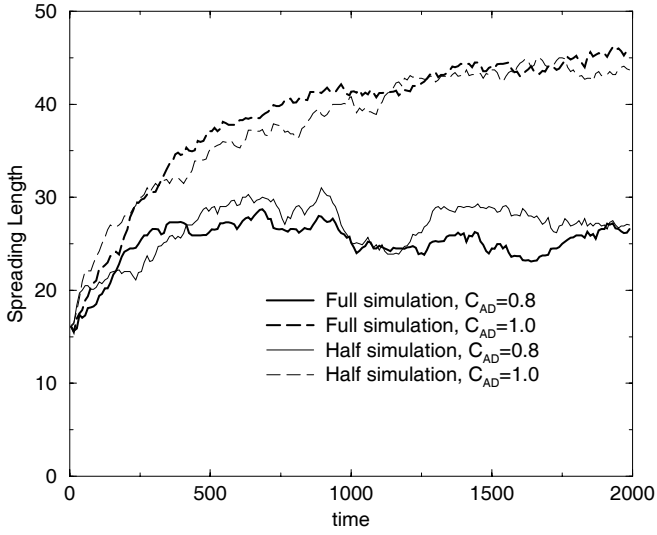


Fig. 5. Comparison of half drop and full drop simulations. The "spreading length" is the distance from the edge of the drop to its midpoint. IT is measured by inspecting the density of fluid atoms just above the solid surface.

3 The Effect of Surfactants

We next consider the effect of surfactant composition on spreading. A half drop of 4500 molecules of A_2 fluid was prepared and 90 molecules of C_3B_3, C_2B_4 or CB_5 were added. We placed each drop on a solid surface, and measured its spreading. In Fig. 6a, we compare the results against pure fluid drops with the same mass (4770 molecules of A_2). The surfactants do enhance the spreading, but not by much. Furthermore, it is not clear which surfactant works best. Looking at a snapshot of the simulation, we see that the surfactants are concentrated both at the free surface and at the solid-liquid interface, as shown in Fig. 6b.

The surfactants do partially shield the solvent from the surface, but they also interpose the hydrophobic heads between the solvent and the surface. The

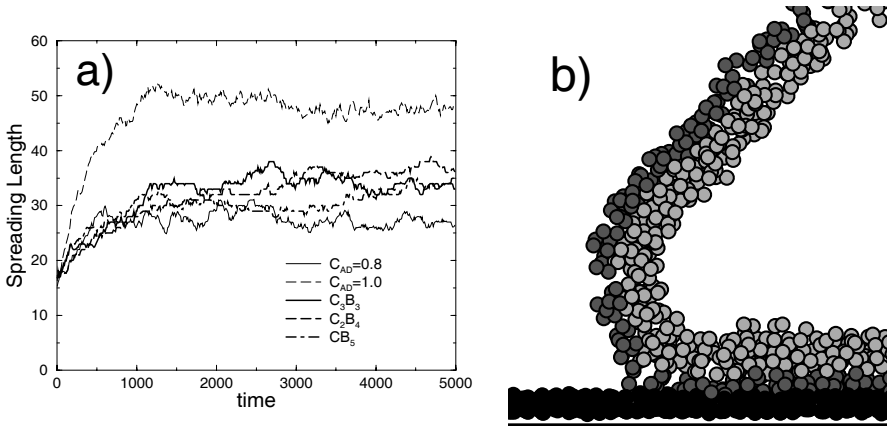


Fig. 6. a) Comparison of different surfactants. The thin lines give pure fluid spreading for two different values of C_{AD}. The surfactant drops have $C_{AD} = 0.8$; the presence of surfactants does enhance spreading, but not as much as increasing C_{AD} to 1. The interaction coefficients are as given in Sec. 2.1: $C_{AA} = C_{BB} = C_{DD} = C_{CD} = 1$, $C_{AC} = C_{BD} = C_{CC} = 0$, and $C_{AB} = 2$. b) A magnified view of the edge of the spreading drop at the end of the C_2B_4 simulation in a). The fluid atoms are not shown.

presence of the hydrophobic heads at the liquid-solid boundary reduces spreading because the hydrophobic heads repel the liquid.

Making the surfactant head less hydrophobic might improve the spreading, because the hydrophobic heads at the solid-liquid boundary would be less repulsive to the liquid. We changed the fluid-surfactant interaction parameters to $C_{AC} = 0.5$ and $C_{AB} = 1.5$, but left all the others unchanged. As a result, the surfactant becomes soluble, that is, it is not confined to the exterior of the drop, and some molecules are present in the interior. However, the majority of the surfactant molecules remain at the surface. The comparison of the soluble and insoluble C_2B_4 surfactant is shown in Fig. 7. As one can see, the soluble surfactant is not more successful than the insoluble one.

The failure of the soluble surfactants to further enhance spreading casts doubt on the "shielding" mechanism. A surfactant must have a hydrophobic part, and this part must always be trapped between the fluid and the solid. If the hydrophobic heads prevent successful spreading even when they are only mildly hydrophobic, it is difficult to see how any surfactant will increase spreading. But if the attraction between the hydrophobic part of the surfactant and the solid surface is increased, the spreading increases remarkably, as shown in Fig. 8.

In Fig. 8b, we see the reason for this surprising behavior: the hydrophobic head penetrate into the solid surface. In this way, they are hidden from the fluid atoms, and the spreading of the drop is much enhanced.

Note that the increased spreading is due to the shielding mechanism, not to a reduction in surface tension. The fluid-surfactant interaction has been left

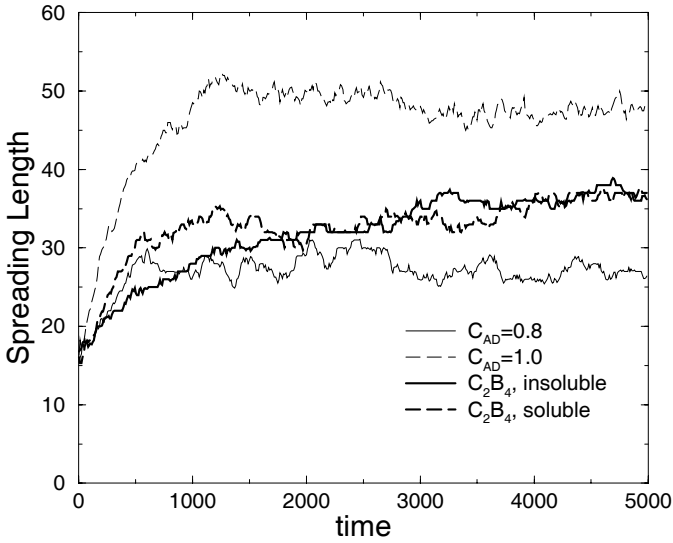


Fig. 7. Comparison between soluble and insoluble surfactants. In both cases, the surfactant is C_2B_4. The insoluble case is taken from Fig. 6, and the interaction coefficients are as stated in Sec. 2.1. The soluble case is the same, except two interaction coefficients have been modified: $C_{AC} = 0.5$ and $C_{AB} = 1.5$.

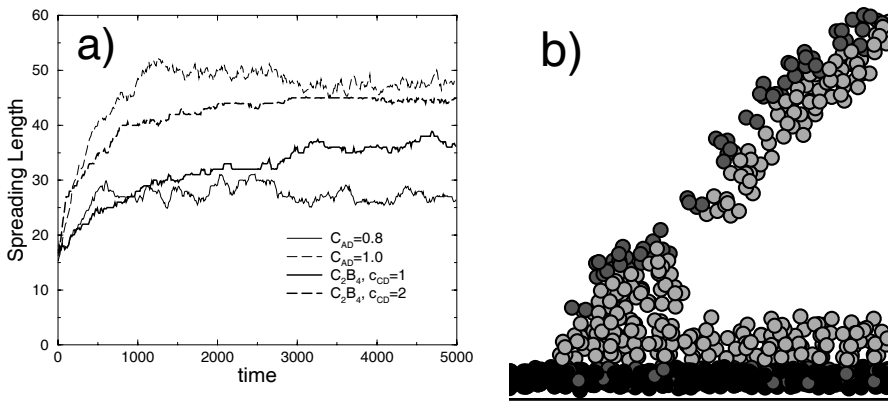


Fig. 8. a) Comparison between the standard surfactants (of Fig. 6) and surfactants with enhanced attraction to the solid. In this last case, $C_{CD} = 2$, with all the interaction coefficients as in Sec. 2.1. b) A magnified view of the edge of the spreading drop molecules at the end of the C_2B_4 simulation in a). The fluid atoms are not shown. In contrast to Fig. 6b, the hydrophobic heads penetrate into the solid. This is because the interaction coefficient C_{CD} has been increased from 1 to 2.

unchanged, so tension of the free surface will not change. (If anything, the surface tension will increase, because the concentration of surfactant molecules there is lower in Fig. 8b than in Fig. 6b.) We have therefore demonstrated that the shielding mechanism is active.

4 Conclusions

We have studied spreading drops using molecular dynamics simulations. We have presented novel boundary conditions which permit us to double the size of the studied drop. We also showed that surfactants enhance spreading. The performance of most surfactants is limited by the fact that the hydrophobic head becomes trapped between the solid and the liquid. Since the hydrophobic head repels the liquid, the liquid soon stops spreading. However, if the attraction between the hydrophobic head and the solid is strong enough, the hydrophobic head buries itself into the solid, thus hiding itself from the liquid, and spreading is greatly enhanced.

Our work suggests that the "shielding" spreading mechanism can be important, if the hydrophobic heads can be hidden from the fluid. In our case, this is acheived by hiding the heads in the solid itself. But more realistic molecules could hide the heads by other mechanisms, such as an orderly array of surfactants at the solid-liquid boundary.

Acknowledgements

We thank A. Couzis and C. Maldarelli for discussions, the NASA Microgravity Program for financial support, and NCCS and NPACI for computer resources.

References

1. T. Stoebe, Z. Lin, R. M. Hill, M. D. Ward, and H. T. Davis, Langmuir **12**, 337 (1996) and **13**, 7282 (1997); M. J. Rosen and L. D. Song, Langmuir **12**, 4945 (1996); T. Svitova et al., Langmuir **14**, 5023 (1998).
2. A. W. Adamson and A. P. Gast, *Physical chemistry of surfaces*, 6th ed., (Wiley, New York, 1997).
3. B. Smit, Phys. Rev. A **37**, 3431 (1988); F. Schmid, in *Computational Methods in Surface and Colloid Science*, Surface Science Series **89**, 631 (2000).
4. U. D'Ortona, J. De Coninck, J. Koplik, and J. Banavar, Phys. Rev. E, **53** 562 (1996).
5. M.P. Allen and D.J. Tildesley, *Computer Simulation of Liquids*, (Oxford, Clarendon Press, 1987).

Supporting Car-Parrinello Molecular Dynamics with UNICORE

Valentina Huber

Central Institute for Applied Mathematics, Research Centre Jülich,
Leo-Brandt-Str, D-52428 Jülich,
Germany,
v.huber@fz-juelich.de

Abstract. This paper presents the integration of application specific interfaces in the UNICORE Grid infrastructure. UNICORE provides a seamless and secure mechanism to access distributed supercomputer resources. The widely used Car-Parrinello Molecular Dynamics (CPMD) application was selected as a first example to demonstrate the capabilities of UNICORE to scientists. Through the graphical interface, developed at Research Centre Jülich, the user can prepare a CPMD job and run it on a variety of systems at different locations. In addition, the "*CPMD Wizard*" makes it easy to configure the full set of the control parameters, cell properties, pseudopotentials and atom positions for the CPMD simulation.

1 Introduction

UNICORE (**UN**iform **I**nterface to **CO**mputer **RE**sources) [4] provides a science and engineering Grid [10] combining resources of supercomputer centers and makes them available through the Internet. The UNICORE user benefits from the seamless access through the graphical *UNICORE Client* to the distributed resources to solve large problems in computational science without having to learn about the differences between execution platforms and environments.

One important success criterion for UNICORE is the integration of already existing applications. We selected the widely used Car-Parrinello Molecular Dynamics code [1] as a first application to be integrated in UNICORE. CPMD is an *ab initio* Electronic Structure and Molecular Dynamics program; since 1995 the development is continued at the Max-Planck Institut für Festkörperforschung in Stuttgart [3]. This application uses a large amount of CPU time and disk space and is the ideal candidate for a GRID application. Currently, multi processor versions for IBM Risc and Cray PVP systems and parallel versions for IBM SP2 and Cray T3E are available.

Presently a wide variety of groups and projects [11] are experimenting with Grid applications and middleware: Globus [12], Legion [13], WebFlow [14], WebSubmit [15], HotPage [16], Teraweb [17]. They provide simple interfaces that allow users to select an application, select a target machine, submit the job, and monitor the job's progress. Our approach goes far beyond the simplistic functionality just mentioned. The new graphical CPMD user interface uses the

V.N. Alexandrov et al. (Eds.): ICCS 2001, LNCS 2073, pp. 560–566, 2001.
© Springer-Verlag Berlin Heidelberg 2001

standard functions of UNICORE for authentication, security and data transfer [9]. Furthermore the interface provides the users with a intuitive way to specify the full set of configuration parameters (specification of the input and output files, the library for pseudopotentials, etc.) for a CPMD simulation. In addition it allows to run the CPMD simulations, which comprise many steps in a pipeline. Thes steps, or "tasks", include importing, preprocessing and in many cases, several tasks in sequence (with data-flow type dependencies) must be executed to obtain the final results.

2 The CPMD Wizard

The input file required for CPMD is composed of different sections, contains over 150 different keywords, and has a complex format [2]. To prepare a correct CPMD job the user has to know the internal structure of the CPMD input in detail. A *CPMD Wizard* has been integrated to assist the user. It is started within the *UNICORE Client*.

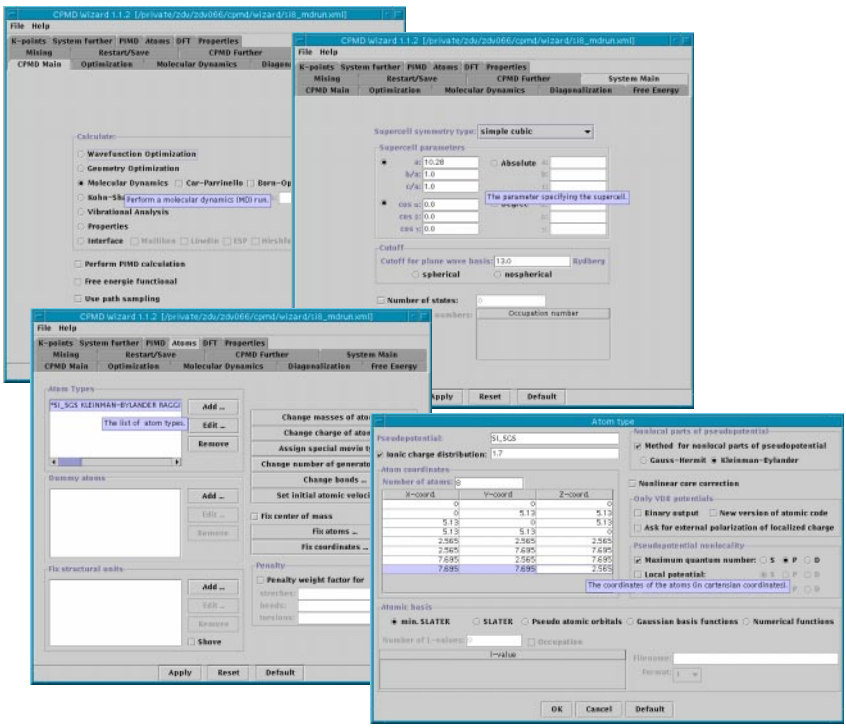


Fig. 1. CPMD Wizard generates the CPMD input automatically.

The *CPMD Wizard* is a graphical interface, implemented as a Java-2 application, which allows the user to generate the CPMD specific parameters auto-

matically. It is composed of different panels matching the structure of different sections in the CPMD input file: *CPMD Main* specifies control parameters for calculation, *Optimization* - optimization parameters, *Diagonalization* - diagonalization schemes, *System Main* - information about the supercell, *Atoms* - atom positions and pseudopotentials, etc. (see Fig. 1).

The interface provides descriptions for each option, that pop up when the mouse lingers over a field. Based on the context, e.g. previously selected options, it allows or prevents input to depended fields. The inactive fields are indicated with a shadow color.

The Wizard prompts the user for missing mandatory data or to correct data that does not match the format specification. It uses XML as its internal data format to facilitate parsing and validation of CPMD input.

3 Preparation of CPMD Jobs

Creation of the input file is one of the tasks, which is greatly simplified by the *CPMD Wizard*. In addition, the CPMD job must contain resource specifications of input and output data sets that the CPMD application expects. A customized graphical interface, using standard UNICORE functions, guides the user.

Fig. 2 shows the input panel for one CPMD task, in this case a molecular dynamics run. It is divided into four areas: *Properties*, the configuration area for the CPMD calculation, data *Imports* and data *Exports*.

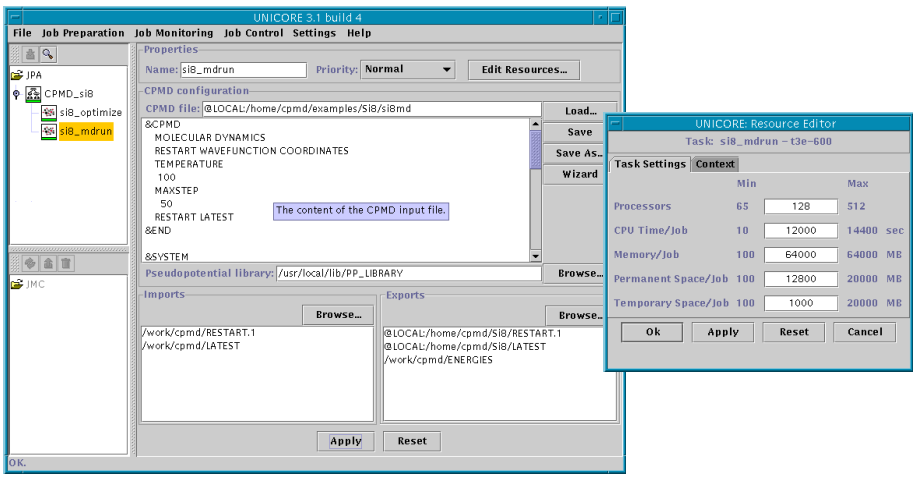


Fig. 2. GUI for the CPMD task.

The *Properties* area contains global settings like the task name, the task's resource requirements and the task's priority. The resource description includes the number of processors, the maximum CPU time, the amount of memory, the

required permanent and the temporary disk space. The Job Preparation Agent (*JPA*), part of *UNICORE Client*, knows about the minimum and the maximum values for all resources of the execution system, where the task is to be run, and incorrent values are shown in red.

The configuration area shows the data generated by the CPMD Wizard. Experienced users may use the data from existing jobs, stored on the local computer. The configuration data may be edited directly or through the Wizard. It is also possible to save data as a text file on the local disk.

For all atomic species, which will be used in the CPMD calculation, the path to the pseudopotential library has to be specified. The local pseudopotential files will be automatically transferred to the target system. Alternatively, the user can specify the remote directory for the pseudopotentials. If this field is empty, then the default library on the destination system will be used.

The *Imports* area lists the set of input files for the CPMD calculation, e.g. a restart file to reuse the simulation results from a previous step. The input files may reside on the local disk or on the target system. Local files are automatically transferred to the target system and remote files will be imported to the job directory.

The *Exports* area controls the disposition of the result files to be saved after the the job completion. In the example some of the output files will be stored on the target system and others, marked *@LOCAL*, will be transferred to the local system and can be visualized there.

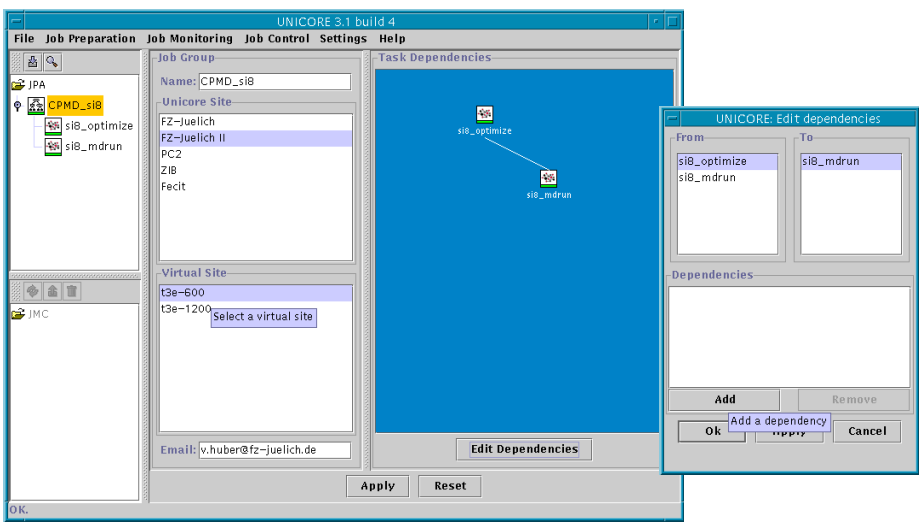


Fig. 3. CPMD job consisting of two tasks and dependency between them.

Fig. 3 represents an example of a CPMD job consisting of two steps: *si8_optimize* task for the wavefunction optimization of a cluster of 8 Silicon atoms and

si8_mdrun task for molecular dynamics run. Both tasks will be executed on the same system, T3E in Jülich. The left hand side of the JPA represents the hierarchical job structure. The green color of the icons indicates the job as *Ready for submission*. The second task will be run only after the first one is completed. It uses the output files from the *si8_optimize* task to reuse the results of the wavefunction optimization. This dependency is shown on the right hand side and represents a temporal relation between the tasks.

Before the CPMD job can be submitted to a particular target system, the interface automatically checks the correctness of the job. Prepared jobs can be stored to be reused in the future.

UNICORE has all the functions to group CPMD tasks and other tasks into jobs. Each task of a job may execute on a different target host of the UNICORE Grid. UNICORE controls the execution sequence, honoring dependencies and transfers data between hosts automatically.

4 Monitoring of CPMD Jobs

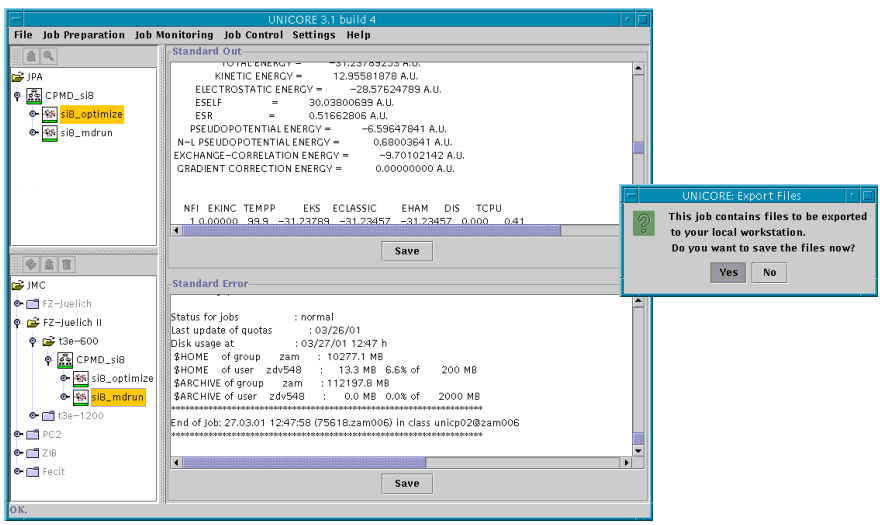


Fig. 4. The Job monitor displays the status of the jobs submitted to a particular system.

The user can monitor and control the submitted jobs using the job monitor part of the UNICORE Client. The job monitor displays the list of all jobs the user has submitted to a particular system. The job, initially represented by an icon, that can be expanded to show the hierarchical structure. The status of jobs or parts of jobs are given by colors: green - completed successfully, blue -

queued, yellow - running, red - completed not successfully, etc. It is possible to terminate running jobs or to delete the completed job from the list of jobs.

After a job or a part of a job is finished, the user can retrieve its output.

Fig. 4 presents the status of the jobs submitted to the *T3E* system in Jülich. The right hand side displays the summary standard output and standard error from two steps *si8_optimize* and *si8_mdrun* of *CPMD_si8*.

5 CPMD Integration into UNICORE

Support for the application specific interfaces is based on the "plug-in concept" of the UNICORE Client. Fig. 5 presents the dialog for the setting of user defaults, where the user can specify the plug-in directory for the applications.

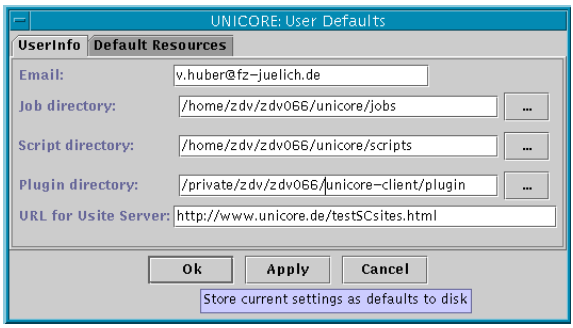


Fig. 5. User settings for the application plug-in directory.

The UNICORE Client scans this directory for the classes implementing the *IUnicorePlugable* interface. In the case of CPMD it is the class *CPMD_Plugin*. The *CPMD_Plugin* adds a new option "Add CPMD" to the menu of the JPA and provides methods to display the CPMD GUI in the UNICORE Client. Other classes required for the CPMD integration are:*CPMD_JPAPanel* and *CPMD_Container*.

Fig. 6. Basic classes for the CPMD integration.

The *CPMD_JPAPanel* represents the GUI for the CPMD application and provides the methods to store the data in the *CPMD_Container*.

The *CPMD_Container* keeps the actual CPMD data, used for information exchange between the *CPMD_JPA* and *Client*, which submits the jobs. It checks for correctness of the input data and builds the internal graph of the CPMD task including all required dependencies. In addition, the *CPMD_Container* provides the application specific icon for the *Client*.

Fig. 6 presents the relationship between CPMD classes and *UNICORE Client*.

6 Outlook

The technigue used for the CPMD integration is extensible to numerous other applications. We plan in such a way to develop the interfaces for MSC-NASTRAN, FLUENT and STAR-CD applications. These interfaces are going to be integrated into *UNICORE Client* for seamless submitting and controlling of jobs.

In the future it is planned to build a generic interface to allow easier integration of applications.

References

1. Marx, D., Hutter, J.: Ab *Initio* Molecular Dynamics: Theory and Implementation Modern Methods and Algorithms of Quantum Chemistry (2000) 329–478
2. Hutter, J.: Car-Parrinello Molecular Dynamics - An Electronic Structure and Molecular Dynamics Program. CPMD Manual (2000)
3. Research Group of Michele Parrinello - http://www.mpi-stuttgart.mpg.de/parrinello
4. UNICORE Project - `http://www.fz-juelich.de/unicore`
5. UNICORE Forum e.V. - `http://www.unicore.org`
6. J. Almond, D.Snelling: UNICORE: uniform access to supercomputing as an element of electronic commerce. FGCS **15** (1999) 539-548
7. J. Almond, D.Snelling: UNICORE: Secure and Uniform access to distributed Resources via World Wide Web. A White Paper. `http://www.kfa-juelich.de/zam/RD/coop/unicore/whitepaper.ps`
8. Romberg, M.: UNICORE: Beyond Web-based Job-Submission. Cray User Group Conference (2000)
9. Romberg, M.: The UNICORE Grid Infrastructure. SGI'2000 Conference (2000)
10. Foster, I. and Kesselman, C. (editors), The Grid: Blueprint for a Future Computing Infrastructure, Morgan Kaufmann Publishers, USA, 1999
11. Global Grid Forum - `http://www.gridforum.org`
12. Globus: The Globus Grid Computing Toolkit - `http://www.globus.org`
13. Legion - `http://legion.verginia.edu`
14. Web Flow: Web Based Metacomputing - `http://www.npac.syr.edu/users/haupt/WebFlow`
15. WebSubmit: A Web Interface to Remote High-Performance Computing Resources `https://www.itl.nist.gov/div895/sasg/websubmit/websubmit.html`
16. HotPage - `http://hotpage.npaci.edu`
17. Teraweb - `http://www.arc.umn.edu/structure/`

Parallel Methods in Time Dependent Approaches to Reactive Scattering Calculations

Valentina Piermarini[1], Leonardo Pacifici[1], Stefano Crocchianti[1], Antonio Laganà[1], Giuseppina D'Agosto[2], and Sergio Tasso[2]

[1] Dipartimento di Chimica, Università di Perugia, Via Elce di Sotto, 8, 06123 Perugia, Italy
[2] Centro Ateneo Servizi Informatici, Università di Perugia, 06123 Perugia, Italy

Abstract. The possibility of implementing a suitable model to parallelize time-dependent approaches to the calculation of quantum reactive probabilities of elementary atom diatom processes based on collocation methods is investigated. Problems arising when adopting a coarse grain model by distributing fixed total angular momentum quantum number J calculations are discussed. A comparison is made with finer grain models based either on a domain decomposition due to the distribution of the various projections Λ of J or to a partitioning of the collocation matrices among the available processors.
Measurements performed indicate that a finer grain parallelism is a proper solution if the communications can be confined into reasonable limits. Otherwise, fine grain parallelism is only an "extrema ratio" for dealing with problems based on matrix representations too large to be dealt by a single processor.

1 Introduction

Parallelization of computer codes devoted to the calculation of the properties of reacting chemical systems has received increasing attention in recent years [1]. This is motivated by the need for building realistic molecular simulations of several processes of interest for environmental monitoring and control, technology and material research and development, drug and enzyme design.

Up to now this has been the realm of molecular mechanics and classical dynamics that is of approaches based upon the assumption that atoms behave as mass points or microscopic solids moving on a potential energy surface according to the laws of classical mechanics. More recently, especially for a few atom systems, progress has been made in building computational tools based on the more rigorous quantum mechanical assumption that atoms and molecules need to be represented by wavefunctions.

Quantum mechanical calculations, however, are more difficult to carry out than classical ones. They, in fact, require large memories to store the information about the basis set (and related integrals, eigenvalues, eigenvectors, etc.) in which the wavefunction of the molecular system being considered has been expanded.

V.N. Alexandrov et al. (Eds.): ICCS 2001, LNCS 2073, pp. 567–575, 2001.
© Springer-Verlag Berlin Heidelberg 2001

This technique is largely preferred in time independent quantum approaches. As an alternative, the wavefunction of the molecular system can be represented as values on a grid (collocation methods). This technique is preferred for time dependent quantum approaches. In this case, larger memories are needed to store the values of the wavefunction at the chosen grid points.

In collaboration with other laboratories, we have developed computational procedures able to describe reactions and calculate related observables. These procedures are highly time consuming and have been analysed for parallelization. Extensive work has been carried out to parallelize computer codes based on time independent quantum approaches. In particular, both task farm and pipe-line models have been considered for the implementation of full [2] and reduced dimensionality[3] computational procedures on parallel architectures.

On the contrary, only limited work has been performed for parallelizing time dependent approaches. Aim of this paper is to describe the implementation of some parallel models for computer codes based on time dependent methods. In section 2 a brief description of the time dependent approach is given. In section 3 a coarse grain parallelization scheme is discussed. In section 4 progress towards a fine grain parallelization scheme is illustrated.

2 The Quantum Time Dependent Computational Procedure

The case considered in this paper is the atom-diatom reaction:

$$\mathrm{A} + \mathrm{BC}(v, j) \longrightarrow \mathrm{AB}(v', j') + \mathrm{C} \tag{1}$$

where the reactant diatomic molecule BC is in its vibrotational (vj) state and the product diatomic molecule AB or AC is in its vibrotational $(v'j')$ state (we use primed quantities for products, unprimed for reactants). The time-dependent method used in this work makes use of a grid representation of the wavepacket for distance coordinates and a basis set expansion for the angular coordinate. The propagation in time of the wave packet considers only its real part [4,5].

At the very beginning ($t = 0$), for a given value of the total angular momentum quantum number J and its projection Λ on the z axis of a body fixed frame, the system wavefunction $\Psi^{J\Lambda}$ is defined as

$$\Psi^{J\Lambda}(R, r, \Theta; t) = \left(\frac{8\alpha}{\pi}\right)^{\frac{1}{4}} e^{-\alpha(R-R_0)^2}$$

$$\cdot\, e^{-ik(R-R_0)} \cdot \varphi_{vj}^{BC}(r) \cdot P_j^{\Lambda}(\Theta). \tag{2}$$

where R, r and Θ are the Jacobi internal coordinates of the reactant atom diatom system. In eq. 2, $e^{-ik(R-R_0)}$ is a phase factor which gives the initial wave packet a relative kinetic energy towards the interaction region, $\varphi_{vj}^{BC}(r)$ is the initial diatomic molecule BC wavefunction (for the vibrational state v and the rotational state j) expressed in the Jacobi coordinates of the reactant arrangement, $P_j^{\Lambda}(\Theta)$ is the normalised associated Legendre polynomial and k

is the wavevector which determines the relative kinetic energy of the collisional partners [4]. In this way, the wavefunction is defined for a given accessible state of the reactants and a given collisional energy range.

To move the wavepacket out of the reactant region, the integration in time of the time-dependent Schrödinger equation

$$i\hbar\frac{\partial}{\partial t}\Psi^{J\Lambda}(R,r,\Theta;t) = \hat{H}\Psi^{J\Lambda}(R,r,\Theta;t) \tag{3}$$

is performed. The hamiltonian $\hat{H}$ consists of a kinetic part ($\hat{T}$) and a potential part (V) which are multiplicative in the momentum and in the coordinate space, respectively. To exploit this property the application of $\hat{H}$ on Ψ is performed by switching from the coordinate to the momentum space and viceversa for each coordinate. This is the most demanding part of the code in terms of computing time. In fact, this requires that a back and forth Fourier trasform of the wavefunction for all the coordinates involved is repeated at each time step. As an alternative, one can apply a discrete variable (DVR) method for which there is no need to perform the fast Fourier transform although interaction of neighbouring elements of the wave function need to be considered.

In order to analyse the dynamics of the reactive process we require the wavefunction to be expanded in terms of the final diatomic molecule AB wavefunction ($\varphi^{AB}_{v'j'}(r')$ where r', R' and Θ' are the Jacobi coordinates of the product arrangement). From the cut of the wavepacket at the analysis line located far away in the asymptotic region, one can evaluate the time dependent coefficients $C^{J}_{vj\Lambda,v'j'\Lambda'}(t)$ of the expansion. By half Fourier transforming the time-dependent coefficients $C^{J}_{vj\Lambda,v'j'\Lambda'}(t)$ one obtains a set of energy-dependent coefficients whose square modulus is the reaction probability.

The computer code performing the propagation in time of the wavepacket is TIDEP. It carries out iteratively the propagation of the real part of the wavefunction and stores at each time step the value of the $C^{J}_{vj\Lambda,v'j'\Lambda'}(t)$ coefficients. The analysis of the coefficients to work out reaction probabilities is performed off line by another program called TIDAN.

3 The Coarse Grain Parallelization

Parallelization efforts were concentrated on TIDEP since this program is the most time consuming component of the computational procedure (the propagation step has to be iterated for about $10^4 \div 10^5$ times). The structure of TIDEP is:

Read input data: v, j, k, masses, ...
Perform preliminary calculations
LOOP on J
  LOOP on t
    LOOP on Λ

```
            Perform time step integration
            Perform the asymptotic analysis
            Store C(t) coefficients
        END loop on Λ
    END loop on t
END loop on J
Calculate final quantities
Print outputs
```

As can be seen from the scheme given above, calculations are performed at a given range of energy, at a fixed value of the vibrotational quantum number (vj) of the reactant diatom and a single value of the total angular momentum quantum number J. Therefore, the coarsest grain of parallelism that can be adopted is the one distributing the calculation for a given vibrotational state, a given interval of translational energy and a fixed value of J. In this case, the characteristics of the various tasks are so different that a task farm dynamically assigning the computational workload can be adopted. This very coarse grain approach was fruitfully implemented on a cluster of powerful workstations.

To carry out the calculations the value of the physical parameters was chosen to be that of the $O(^1D)$+HCl atom diatom reaction[5]. Accordingly, the mass values were chosen to be 15.9949 amu for O, 1.00783 amu for H atom and 34.96885 amu for Cl. The energy range covered by the calculation was approximately 1 eV, the initial vibrotational state used for the test was $v = 0$ and $j = 0$. The potential energy surface used for the calculations is described in ref. [5] where other details are also given. Two types of gridsize were used for R' and r' (the angular part was in both cases expanded over 80 basis functions): (a) 127 $\times$ 119 points; (b) 251 $\times$ 143 points. Time propagation iterates for about 40000 steps to properly diffuse the wavepacket at $J = 0$. Production runs take about 3 weeks on a Sylicon Graphics PowerChallenge supercomputer. This means also that to calculate a state to state cross section or, even worse, a state to state rate coefficient, the amount of time needed to perform the calculation goes beyond any acceptable limit if simplifications are not introduced. In fact, to evaluate a vibrational state selected rate coefficient, the calculation needs to be for all the reactant rotational states j populated at the temperature considered. In addition, the calculations need to be converged with J and convergence is usually reached only at $J > 100$. This increases enormously the computational load not only because calculations have to be repeated for all J values but also because the dimension of the matrices to be handled in a single J calculation is $J + 1$ times larger than that of $J = 0$. As a matter of fact, the computing time, that depends on the third power of the matrix dimension, rapidly becomes exceedingly large even at small J values. This makes the calculation unfeasible on the machines presently available for academic use at the large scale computing facilities in Europe.

For the same reason this parallel model is even less applicable to four or more atom reactions.

4 The Finer Coarse Grain Parallelization

A next lower level of parallelization is the one based on the combined distribution of fixed J and fixed Λ calculations. As it has been already shown above, there is no problem in distributing fixed J calculations: J is a good quantum number (*i.e.* calculations for different J values are fully decoupled) and, accordingly, the parallelization on J is natural. On the contrary, the decoupling of Λ is not natural since one has to introduce physical constraints of the centrifugal sudden type (*i.e.* the projection of J on the z axis of the body fixed frame remains constant during the collision). This allows to perform separately the step-propagation of the wavepacket for blocks of fixed Λ values and the recombination of the various contributions only at the end of the propagation step. This is a key feature of the adopted computational scheme since it allows a decomposition of the domain of the wavepacket that otherwhise would lead to a drastic increase of the demand for memory when J increases.

Such a parallel model was first tested [6] on the Cray T3E of EPCC (Edinburgh, UK) for the simplest case of $J = 0$ and $J = 1$. In this case, only three pairs of J and Λ values needed to be considered and only three nodes were used (this is the smallest non zero J calculation allowing an investigation of the model proposed). To further save on computing time the integration process was truncated after a few steps (computing time depends linearly on the number of propagation steps and their reduction leads only to a slight underestimate of the speedup). Measured speedups are 2.6 and 2.5 for propagation grids (a) and (b), respectively. This clearly indicates that the proposed model is quite effective in reducing computing times. However, for this parallel model I/O is a real bottleneck, since as reported in ref. [6], it accounts for about 1/5 of the overall computing time. Moreover, I/O times of node zero (through which all I/O is channeled) are four order of magnitude larger than those of the workers. This clearly indicates that there is still room for improvement. In fact, by entirely conveying I/O through the master node, one has the advantage of simplifying I/O operations at the expenses of overloading the master node.

Accordingly, when generalizing the model to higher J values node zero was exclusively dedicated to acting as a master and the centralized management of I/O was abandoned. On the contrary, the feature of carrying out fixed J calculations in pairs (including all the $J + 1$ component of Λ) was kept. The fact that the parallelization is performed on J sets some limits to the maximum value of the total angular momentum quantum number that can be handled by the program. Using this model, in fact, the maximum value of J has to be 3 units lower than the number of processors. In addition, in order to keep all the processors busy, the pairs of J values simultaneously running have to sum up to the maximum allowed value. As an example, for a 16 node machine the maximum allowed value of J is 13. In this case, in fact, one node acts as a master, at least one node is reserved to the smaller J calculation and the remaining ones to the larger J calculation. Their values are chosen so as to make the sum equal to 13 (the number of Λ values is $J + 1$ since it varies from 0 to J). The looping on J starts from the pair $J = 0$ (one Λ value) and $J = 13$ (fourteen Λ values) and

then it continues by rising the lower J value and lowering the higher one until the ascending and descending sequences converge.

To evaluate the performances of the model, the calculations were performed on the Origin 3800 of Cineca (Bologna, I) using the same set of parameters adopted for the tests described above (grid (a)) and reducing the basis set expansion for the angular part to 10. Measured times are shown in Table 1 where the average node computing time (in second) is given for different J values.

Table 1. Execution time in seconds

J	0	1	2	3	4	5	6	7	8	9	10	11	12	13
time/s	4400	4440	4480	4540	4680	4680	4760	5080	5160	5120	5240	5280	5360	5400

As clearly shown by the results reported in the Table, the computing time per node (averaged over the various values of Λ) increases with J. This indicates that the increase of communication time associated with an increase in the number of allowed Λ values penalizes the efficiency of the code. However, its entity is small and the model shows to be quite effective in reducing the overall computing time.

5 Fine Grain Parallelization

Further attempts have been made to evaluate the possibility of pushing the parallelization to a very fine granularity. This turns out to be useful either when a specific J value calculation needs to be carried out or when the dimensionality of the problem becomes so large (like in the case of polyatomic reactions) that it does not fit into the individual node memory. This particularly applies when computer center policies set severe limitations on memory and on the amount of computing time assigned to individual jobs.

To work out a finer grain parallel model the fixed angle version of the code making use of the DVR technique to perform the propagation was used. Related operations are performed inside the routine av. Inside av two matrix times vector and one vector times vector operations are performed according to the following computational scheme:

LOOP of iv from 1 to nv
  LOOP of ir from 1 to nr
    a(ir,iv) = 0
  END loop of ir
END loop of iv
LOOP of iv from 1 to nv
  LOOP of i from 1 to nr

```
        LOOP of ip from 1 to nr
            a(i,iv) = a(i,iv) + b(i,ip)*c(ip,iv)
        END loop of ip
    END loop of i
END loop of iv
LOOP of i from 1 to nr
    LOOP of iv from 1 to nv
        LOOP of ivp from 1 to nv
            a(i,iv) = a(i,iv) + s(iv,ivp)*c(i,ivp)
        END loop of ivp
    END loop of iv
END loop of i
LOOP of iv from 1 to nv
    LOOP of i from 1 to nr
        a(i,iv) = a(i,iv) + v(i,iv)*c(i,iv)
    END loop of i
END loop of iv
```

When all the matrices involved are distributed per (groups of) columns among a certain number of nodes, all the operations sketched above imply a quite significant amount of communication to allow the nodes have the updated version of the matrices involved. Test runs were performed on the IBM SP2 of the computer Center of the University of Perugia. For the measurements use was made of 5 processors of which one acts as a master and runs the main program, transfers the information from and to the nodes, combines the pieces of information arriving from the nodes in a form suitable for redistribution and further manipulation. The ratio s/p between measured sequential (single node) and parallel (five nodes) execution times is given in Table 2 for different values of the collocation matrix dimension. The Table clearly shows that the parallel

Table 2. Ratio of sequential/parallel execution time

Dimension	80	400	500	600
s/p	9.5	1.2	1.0	0.98

code is outperformed by the sequential one for small dimensions of the matrices. Break even occurs at dimension 500 for which the two versions of the code take the same amount of time. For larger dimensions the parallel code outperforms the sequential one. This means that, according to measurements performed here, there is no advantage in parallelizing the code at this fine level of granularity unless further investigations and deep restructuring or, possibly, the use of clever

parallelization tools [7] allow a significant reduction in communication time. The only convenience can be found in the fact that a fine grain parallelism allows to deal with matrices too large to be dealt by a single processor memory.

6 Conclusions

The investigation of suitable models for the parallelization quantum time dependent approaches to the calculation of reactive probabilities of elementary atom diatom processes has been carried out for approaches based on collocation methods. The various models investigated show that in order to make the computing time manageable the parallelization has to be pushed to a fairly low level. This has been exploited for time dependent approaches to chemical reactivity by distributing the propagation on time of the fixed angular momentum and fixed angular momentum projection wavepacket and regaining the coupling after each integration step. The model was found to be quite efficient and to break the calculation into a computational grain small enough to make the program run on a reasonable amount of time. On the contrary, when the parallelization is pushed to a lower level and the computational grain is made finer, the overhead due to communications has shown to be so heavy to make the efficiency of the parallel computational procedure very poor. The choice of such a fine grain parallelism is only justified by the need of tackling a problem having a matrix representation too large to be dealt by a single processor.

References

1. Laganà, A. : Innovative computing and detailed properties of elementary reactions using time independent approaches. Comp. Phys. Comm. **116** (1999) 1–16; Laganà, A., Crocchianti, S., Bolloni, A., Piermarini, V., Baraglia, R., Ferrini, R., Laforenza, D. : Computational granularity and parallel models to scale up reactive scattering calculations. Comp. Phys. Comm. **128** (2000) 295-314
2. Laganà, A., Crocchianti, S., Ochoa de Aspuru, G., Gargano, R., Parker, G.A.: Parallel time independent quantum calculations of atom diatom reactivity. Lecture Notes in Computer Science Vol. 1041. Springer-Verlag, Berlin Heidelberg New York (1995) 361-370; (1995) 361-370; Bolloni, A., Riganelli, A., Crocchianti, S., Laganà, A.: Parallel quantum scattering calculations applied to the dynamics of elementary reactions. Lecture Notes in Computer Science Vol. 1497. Springer-Verlag, Berlin Heidelberg New York (1998) 331-339
3. Baraglia, R., Laforenza, D., Laganà, A.: Parallelization strategies for a reduced dimensionality calculation of quantum reactive scattering cross sections on a hypercube machine. Lecture Notes in Computer Science Vol. 919. Springer-Verlag, Berlin Heidelberg New York (1995) 554–561
4. Balint-Kurti, G. G.: Time dependent quantum approaches to chemical reactivity. Lecture Notes in Chemistry Vol. 75. Springer-Verlag, Berlin Heidelberg New York (2000) 74–87

5. Balint-Kurti, G.G., Dixon, R. N., Marston. C. C.: Grid methods for solving the Shrödinger equation and time dependent quantum dynamics of molecular photofragmentation and reactive scattering processes. International Reviews in Physical Chemistry **111** (1992) 317–344; V. Piermarini, V., Balint-Kurti, G.G., Gray, S., Gogtas, F., Hernandez, M.L., Laganà, A.: Wavepacket calculation of cross sections, product state distributions and branching ratios for the $O(^1D)+$ HCl reaction. J. Phys. Chem (in the press)
6. Piermarini, V., Laganà, A., Smith, L., Balint-Kurti, G. G., Allan, R. J.: Parallelism and granularity in time dependent approaches to reactive scattering calculations. PDPTA **5** (2000) 2879–2884
7. Vanneschi, M.: Heterogeneous High Performance Computing environment. Lecture Notes in Computer Science Vol. 1470.Springer-Verlag, Berlin Heidelberg New York (1998) 21-34

Construction of Multinomial Lattice Random Walks for Optimal Hedges*

Yuji Yamada and James A. Primbs

Control and Dynamical Systems, California Institute of Technology,
MC 107-81, Pasadena, CA 91125, USA,
{yuji, jprimbs}@cds.caltech.edu

Abstract. In this paper, we provide a parameterization of multinomial lattice random walks which take cumulants into account. In the binomial and trinomial lattice cases, it reduces to standard results. Additionally, we show that higher order cumulants may be taken into account by using multinomial lattices with four or more branches. Finally, we outline two synthesis methods which take advantage of the multinomial lattice formulation. One is mean square optimal hedging in an incomplete market and the other involves pricing under "implied volatility" and "implied kurtosis".

1 Introduction

An important issue in pricing and hedging derivatives is the generality of the model for the underlying asset (see e.g., [4,9,10,13]) and its computational tractability. From this standpoint, modeling underlying asset dynamics on a multinomial lattice is useful (see e.g., [5,6,12,14,16,17] and the books of [8,11]) due to the existence of efficient methodologies for solving hedging and pricing problems. Moreover, multinomial lattice techniques allow one to price various types of derivative when no analytical formula is available. This paper seeks to provide a single parameterization for multinomial lattice random walks which can take higher order cumulants into account, instead of only the mean and variance. Before proceeding, we mention that there is an extensive body of literature on the subject of lattice techniques in derivative pricing and hedging, and we hope that readers will excuse our blatant omission of much of that work.

2 Construction of Multinomial Lattices

We will present a general description of a random walk on a multinomial lattice. Consider a stock market in the time interval $t \in [0,\ T]$, where traders are allowed to purchase and sell at discrete times $t_n = n\tau,\ n = 0,\ 1, \ldots, N$, where $\tau := T/N$. Let S_n denote the price of the stock at $t = t_n$, and suppose that u_n and d_n satisfy $u_n > d_n > 0$, then a multinomial tree with L branches at each node is given by

$$S_{n+1} = u_n^{L-l} d_n^{l-1} S_n,\ l = 1, \ldots L \tag{1}$$

* In all correspondence, contact the first author Y. Yamada by E-mail or Fax: +1-626-796-8914.

V.N. Alexandrov et al. (Eds.): ICCS 2001, LNCS 2073, pp. 579–588, 2001.
© Springer-Verlag Berlin Heidelberg 2001

where $p_l,\ l = 1, \dots L$ are the corresponding probabilities which satisfy $p_1 + \cdots + p_L = 1$. To make the multinomial tree recombine, we further assume that $u_n/d_n = c$ for all $n = 0, \dots, N-1$ for some constant $c\ (> 1)$. One can verify that the process in (1) consists of a lattice (or a recombining multinomial tree), where the stock may achieve $n(L-1)+1$ possible prices at time $t = t_n,\ n = 0, \dots, N$. For example, in the case of $u_n = u$ and $d_n = d$ for all $n = 1, \dots, N-1$, the price of the stock at the k-th node from the top of the lattice is given by

$$S_n^{(k)} = u^{n(L-1)+1-k} d^{k-1} S_0,\ k = 1,\ 2, \dots, n(L-1)+1. \tag{2}$$

2.1 Parameterization for Multinomial Lattices with Cumulants

Let X_n be the log stock return between t_n and t_{n+1} defined as

$$X_n := \ln S_{n+1} - \ln S_n,$$

and assume that each X_n is independent. Notice that

$$\ln S_N = \ln S_0 + \sum_{n=0}^{N-1} X_n.$$

We will construct multinomial lattice random walks to model stock price dynamics in terms of (local) cumulants of X_n through suitable choices of the parameters, L, N, u, d and $p_1,\ \dots, p_L$. The m-th cumulant of X_n will be denoted by

$$\mathbb{C}\left(X_n^m\right).$$

Note that the cumulant $\mathbb{C}\left(X_n^m\right)$ is a polynomial in the moments $\mathbb{E}\left(X_n^v\right)$ with $v \le m$, where the first and second cumulants are the mean and variance of X_n, respectively. The third and fourth cumulants are related to skewness and (Fisher) kurtosis, and are given by

$$\mathbb{C}\left(X_n^3\right) = M_n^{(3)},\ \mathbb{C}\left(X_n^4\right) = M_n^{(4)} - 3M_n^{(2)}, \tag{3}$$

where $M_n^{(m)}$ is the m-th central moment given as

$$M_n^{(m)} = \mathbb{E}\left[\left(X_n - \mathbb{E}\left(X_n\right)\right)^m\right].$$

The cumulants have an additive property when independent random variables are summed. For example, the m-th cumulant of $\sum_{n=0}^{N-1} X_n$ is just the sum of the m-th cumulants of X_n for $n = 0, \dots, N-1$.

We will provide a parameterization of multinomial lattice random walks which take cumulants into account. Let

$$u_n := \exp\left(\frac{\nu_n}{L-1} \cdot \tau\right) \exp\left(\sqrt{\frac{\tau}{\alpha}}\right),\ d_n := \exp\left(\frac{\nu_n}{L-1} \cdot \tau\right) \exp\left(-\sqrt{\frac{\tau}{\alpha}}\right) \tag{4}$$

where $\alpha > 0$ is some constant. One can readily see that u_n/d_n is constant for all $n = 0, \ldots, N-1$ if α is fixed. With these choices for u_n and d_n, X_n may be computed as

$$X_n = \ln S_{n+1} - \ln S_n = \nu_n \tau + (L - 2l + 1)\sqrt{\frac{\tau}{\alpha}}.$$

Since we have not specified any variables in (4) yet (except τ $(= T/N)$), we have $L-1$ plus 2 unknown parameters, $p_1, \ldots, p_{L-1}$ (where p_L may be calculated as $p_L = 1 - (p_1 + \cdots + p_{L-1})$, ν_n and α. We will use these parameters to take advantage of additional information (i.e., cumulants).

Suppose that $\nu_n \tau$ is the mean of X_n, i.e., the first cumulant (mean) of X_n is

$$\mathbb{C}(X_n) = \mathbb{E}(X_n) = \nu_n \tau. \tag{5}$$

In this case,

$$\sum_{l=1}^{L} p_l (L - 2l + 1) = \mathbb{E}(L - 2l + 1) = 0, \tag{6}$$

must hold, where the expectation in (6) is taken with respect to $l = 1, \ldots, L$. With this mean value, the m-th central moment $M_n^{(m)}$ is given by

$$M_n^{(m)} = \mathbb{E}\left[(X_n - \nu_n \tau)^m\right] = \left(\sqrt{\frac{\tau}{\alpha}}\right)^m \mathbb{E}\left[(L - 2l + 1)^m\right], \tag{7}$$

and the second through fourth cumulants are computed by $\mathbb{C}(X_n^2) = M_n^{(2)}$ and the formulas in (3).

Binomial Lattice Case: To illustrate the parameterization described above, we first consider the case of $L = 2$, i.e., the binomial lattice case. Since there are already two constraints for the probabilities p_1 and p_2, i.e.,

$$p_1 + p_2 = 1, \quad \sum_{l=1}^{2} p_l (L - 2l + 1) = p_1 - p_2 = 0,$$

we obtain $p_1 = p_2 = 1/2$. Suppose that the variance of X_n is given by $\sigma_n^2 \tau$. This condition restricts $\alpha = 1$ and σ_n to be constant, i.e., $\sigma_n = \sigma$ $(n = 0, \ldots, N-1)$, and we have the binomial lattice formula provided in [11] (see also the original work of [5])

$$u_n = \exp\left(\nu_n \tau + \sigma\sqrt{\tau}\right), \quad d_n = \exp\left(\nu_n \tau - \sigma\sqrt{\tau}\right), \quad p_1 = p_2 = 1/2. \tag{8}$$

Trinomial Lattice Case: In the case of a trinomial lattice, i.e., $L = 3$, we have one more parameter p_3, and this allows us to take local volatility information into account, i.e., the second cumulant. Suppose that the second cumulant (i.e., variance) of X_n is given by $\sigma_n^2\tau$. In this case, we have

$$p_1 + p_2 + p_3 = 1, \quad 2p_1 - 2p_2 = 0, \quad 4p_1 + 4p_2 = \alpha\sigma_n^2. \tag{9}$$

where the second and third equations are obtained from (6) and (7), respectively. By solving (9) with respect to $p_1,\ p_2$ and p_3, we find

$$[p_1,\ p_2,\ p_3] = \left[\frac{\alpha\sigma_n^2}{8},\ 1 - \frac{\alpha\sigma_n^2}{4},\ \frac{\alpha\sigma_n^2}{8}\right].$$

To guarantee that these probabilities are positive, α must satisfy $0 < \alpha < 4/\sigma_n^2$. If σ_n is constant, i.e., $\sigma_n = \sigma\ (n = 0, \ldots, N-1)$, one may use $\alpha = 4/(3\sigma^2)$, which provides a trinomial lattice formula whose up, middle, and down rates and corresponding probabilities are given by

$$u_n^2 = \exp\left(\nu_n\tau + \sigma\sqrt{3\tau}\right), \quad u_n d_n = \exp\left(\nu_n\tau\right), \quad d_n^2 = \exp\left(\nu_n\tau - \sigma\sqrt{3\tau}\right),$$
$$[p_1,\ p_2,\ p_3] = [1/6,\ 2/3,\ 1/6].$$

This also corresponds to a well known finite difference scheme.

If σ_n is a function of $(S_n,\ n)$, i.e., $\sigma_n = \sigma(S_n,\ n)$, the above formula can be modified by writing σ_n in terms of a nominal value $\hat{\sigma}$ as

$$\sigma_n = (1 + \delta_n)\hat{\sigma}.$$

Let α be chosen as $\alpha = 4/(3\hat{\sigma}^2)$. Then the up, middle, and down probabilities are given as

$$[p_1,\ p_2,\ p_3] = \left[\frac{(1+\delta_n)^2}{6},\ 1 - \frac{(1+\delta_n)^2}{3},\ \frac{(1+\delta_n)^2}{6}\right].$$

Note that the probabilities are positive as long as $-\sqrt{3} - 1 < \delta_n < \sqrt{3} - 1$.

Multinomial Lattice Case: Similarly, one can pose additional conditions given by higher order cumulants by using four or more branches in a multinomial lattice. For example, if we have third cumulant information corresponding to skewness, this imposes an additional constraint,

$$\mathbb{C}\left(X_n^3\right) = \left(\sqrt{\frac{\tau}{\alpha}}\right)^3 \mathbb{E}\left[(L - 2l + 1)^m\right] = s_n\tau\left(\sigma_n\sqrt{\tau}\right)^3,$$

where $s_n\tau$ is the skewness of X_n. This condition can be taken into account if four branches are used in the multinomial lattice, i.e., $L = 4$. If we solve four linear equations for the probabilities $p_1,\ p_2,\ p_3,\ p_4$, we obtain

$$[p_1,\ p_2,\ p_3,\ p_4] = \frac{1}{16} \times \left[-1 + \alpha\sigma_n^2\left(1 + \frac{s_n\tau\sqrt{\alpha}\sigma_n}{3}\right),\ 9 - \alpha\sigma_n^2\left(1 + s_n\tau\sqrt{\alpha}\sigma_n\right),\right.$$
$$\left. 9 + \alpha\sigma_n^2\left(-1 + s_n\tau\sqrt{\alpha}\sigma_n\right),\ -1 + \alpha\sigma_n^2\left(1 - \frac{s_n\tau\sqrt{\alpha}\sigma_n}{3}\right)\right].$$

If σ_n is constant, i.e., $\sigma_n = \sigma$ $(n = 0, \ldots, N-1)$, the choice $\alpha = 4/\sigma^2$ results in the following formulas:

$$u_n^3 = \exp\left(\nu_n\tau + \frac{3\sigma}{2}\sqrt{\tau}\right), \quad u_n^2 d_n = \exp\left(\nu_n\tau + \frac{\sigma}{2}\sqrt{\tau}\right),$$

$$u_n^2 d_n = \exp\left(\nu_n\tau - \frac{\sigma}{2}\sqrt{\tau}\right), \quad d_n^3 = \exp\left(\nu_n\tau - \frac{3\sigma}{2}\sqrt{\tau}\right),$$

$$[p_1,\ p_2,\ p_3,\ p_4] = \left[3 + \frac{8}{3}s_n\tau,\ 5 - 2s_n\tau,\ 5 + 2s_n\tau, 3 - \frac{8}{3}s_n\tau\right].$$

If we additionally would like to match the 4th cumulant or "kurtosis", we should introduce a multinomial lattice with five branches, i.e., $L = 5$. Let $\kappa\tau$ denote the kurtosis of X_n, then we need

$$\mathbb{C}\left(X_n^4\right) = \frac{\tau^2}{\alpha^2}\mathbb{E}\left[(6-2l)^4\right] - 3\left(\sigma_n\sqrt{\tau}\right)^4 = \kappa_n\tau\left(\sigma_n\sqrt{\tau}\right)^4,$$

as an additional constraint. In this case, the probabilities $p_1,\ p_2,\ p_3,\ p_4,\ p_5$ can be calculated through the solution of five linear equations, and are given by

$$[p_1,\ p_2,\ p_3,\ p_4,\ p_5] = \frac{1}{96}\left[\alpha\sigma_n^2\left(-1 + s_n\tau\sqrt{\alpha}\sigma_n + \frac{\alpha\sigma_n^2}{4}(3+\kappa_n\tau)\right),\right.$$
$$\alpha\sigma_n^2\left(16 - 2s_n\tau\sqrt{\alpha}\sigma_n - \alpha\sigma_n^2(3+\kappa_n\tau)\right),\ \frac{2}{3}\left\{64 + \alpha\sigma_n^2\left(-20 + \alpha\sigma_n^2\left(3+\kappa_n\tau\right)\right)\right\},$$
$$\left.\alpha\sigma_n^2\left(16 + 2s_n\tau\sqrt{\alpha}\sigma_n - \alpha\sigma_n^2(3+\kappa_n\tau)\right),\ \alpha\sigma_n^2\left(-1 - s_n\tau\sqrt{\alpha}\sigma_n + \frac{\alpha\sigma_n^2}{4}(3+\kappa_n\tau)\right)\right]$$

To understand the effect of kurtosis, assume that $s_n = 0$ and $\sigma_n = \sigma$ $(n = 0, \ldots, N-1)$, then we obtain

$$[p_1,\ p_2,\ p_3,\ p_4,\ p_5] = \frac{1}{96}\left[\alpha\sigma^2\left(-1 + \frac{\alpha\sigma^2}{4}(3+\kappa_n\tau)\right),\right.$$
$$\alpha\sigma^2\left(16 - \alpha\sigma^2(3+\kappa_n\tau)\right),\ \frac{2}{3}\left\{64 + \alpha\sigma^2\left(-20 + \alpha\sigma^2\left(3+\kappa_n\tau\right)\right)\right\},$$
$$\left.\alpha\sigma^2\left(16 - \alpha\sigma^2(3+\kappa_n\tau)\right),\ \alpha\sigma^2\left(-1 + \frac{\alpha\sigma^2}{4}(3+\kappa_n\tau)\right)\right]$$

In this case, all the probabilities are positive if

$$\frac{4}{\sigma^2(3+\kappa_n\tau)} < \alpha < \frac{16}{\sigma^2(3+\kappa_n\tau)}.$$

Furthermore, if we choose $\alpha = 4/\sigma^2$, then the above probabilities reduce to

$$[p_1,\ p_2,\ p_3,\ p_4,\ p_5] =$$
$$\left[\frac{1}{24}\left(2+\kappa_n\tau\right),\ \frac{1}{6}\left(1-\kappa_n\tau\right),\ \frac{1}{4}\left(2+\kappa_n\tau\right),\ \frac{1}{6}\left(1-\kappa_n\tau\right),\ \frac{1}{24}\left(2+\kappa_n\tau\right)\right]. \quad (10)$$

The up-down rates corresponding to five branches can be calculated as

$$u_n^4 = \exp\left(\nu_n\tau + 2\sigma\sqrt{\tau}\right), \quad u_n^3 d_n = \exp\left(\nu_n\tau + \sigma\sqrt{\tau}\right), \quad u_n^2 d_n^2 = \exp\left(\nu_n\tau\right)$$

$$u_n d_n^3 = \exp\left(\nu_n\tau - \sigma\sqrt{\tau}\right), \quad d_n^4 = \exp\left(\nu_n\tau - 2\sigma\sqrt{\tau}\right)$$

We first notice that the probabilities are symmetric, i.e., $p_1 = p_5$ and $p_2 = p_4$. In this formulation, p_1, p_3 and p_5 increase with larger kurtosis. On the other hand, p_2 and p_4 decrease if kurtosis increases. Therefore, this confirms that the probability distribution of X_n becomes heavy tailed under positive kurtosis.

If skewness is not zero, the formulation in (10) becomes

$$[p_1,\ p_2,\ p_3,\ p_4,\ p_5] = \left[\frac{2+\kappa_n\tau+2s_n\tau}{24}, \frac{1-\kappa_n\tau-s_n\tau}{6}, \frac{2+\kappa_n\tau}{4}, \frac{1-\kappa_n\tau+s_n\tau}{6}, \frac{2+\kappa_n\tau-2s_n\tau}{24}\right]$$

with the choice of $\alpha = 4/\sigma^2$. In this case, we readily see that the probabilities are not symmetric if $s_n \not= 0$. Moreover, positive (negative) skewness causesp_1 and p_4 to increase (decrease), and the corresponding probabilities p_5 and p_2 to decrease (increase) by an equal amount.

2.2 Parameterization with Time-Dependent Distributions

In this section, we deal directly with the stock price distribution, rather than characterizing it through cumulants. We will consider the case where the stock price distribution is available at every time t_n. Under this assumption, we show that a multinomial lattice can be constructed as follows:

1. Generate a binomial lattice to match the distribution of the stock at every time step.
2. Create a multinomial lattice based on the binomial lattice.

Let $\mathbb{P}_n(S_n)$ be the probability distribution of the stock at $t = t_n$. $\mathbb{P}_n(S_n)$ may be obtained from historical data.

We begin by using a binomial lattice to describe the stock dynamics. Consider the stock prices arranged on a binomial lattice as shown in the left side of Table 1, where the price of the stock on the k-th node from the top of the lattice is denoted by $S_n^{(k)}$. Furthermore, the probability of obtaining the price S_n^k at time n is given by $P_n^{(k)} = \mathbb{P}_n(S_n^k)$ as shown in Table 1.

Let $p_n^{(k)}$ denote the probability of moving from $S_n^{(k)}$ to $S_{n+1}^{(k)}$ (this corresponds to an "up" move). The probability for a corresponding "down" move from $S_n^{(k)}$ to $S_{n+1}^{(k+1)}$ is given by $p_{n,d}^{(k)} = 1 - p_n^{(k)}$. These probabilities are computed based on the node probabilities $P_n^{(k)}$ $(k = 1, \ldots, N,\ k = 1, \ldots, n+1)$ as follows: Consider the node probabilities at the n-th period, $P_n^{(k)}$ $(k = 1, \ldots, n+1)$, and the node probabilities at the $(n+1)$-th period, $P_{n+1}^{(k)}$ $(k = 1, \ldots, n+2)$, where

Table 1. Stock price and corresponding probability

$$\begin{array}{llllll}
\cdots & S_{N-2}^{(1)} & S_{N-1}^{(1)} & S_N^{(1)} & \cdots & P_{N-2}^{(1)} \; P_{N-1}^{(1)} \; P_N^{(1)} \\
\cdots & S_{N-2}^{(2)} & S_{N-1}^{(2)} & S_N^{(2)} & \cdots & P_{N-2}^{(2)} \; P_{N-1}^{(2)} \; P_N^{(2)} \\
\cdots & S_{N-2}^{(3)} & S_{N-1}^{(3)} & S_N^{(3)} & \cdots & P_{N-2}^{(3)} \; P_{N-1}^{(3)} \; P_N^{(3)} \\
\cdots & S_{N-2}^{(4)} & S_{N-1}^{(4)} & S_N^{(4)} & \cdots & P_{N-2}^{(4)} \; P_{N-1}^{(4)} \; P_N^{(4)} \\
\ddots & S_{N-2}^{(5)} & S_{N-1}^{(5)} & S_N^{(5)} & \ddots & P_{N-2}^{(5)} \; P_{N-1}^{(5)} \; P_N^{(5)} \\
 & & S_{N-1}^{(6)} & S_N^{(6)} & & P_{N-1}^{(6)} \; P_N^{(6)} \\
 & & & S_N^{(7)} & & P_N^{(7)}
\end{array}$$

$n \in [0,\ N-1]$. Since $p_n^{(1)}$ is the probability of obtaining $S_{n+1}^{(1)}$ given $S_n^{(1)}$, it may be calculated as

$$p_n^{(1)} = P_{n+1}^{(1)} / P_n^{(1)}. \tag{11}$$

Similarly, since the probability of obtaining $S_{n+1}^{(k)}$ given $S_n^{(k)}$ satisfies

$$P_{n+1}^{(k)} = \left(1 - p_n^{(k-1)}\right) P_n^{(k-1)} + p_n^{(k)} P_n^{(k)}$$

$p_n^{(k)}$ may be calculated as

$$p_n^{(k)} = \left(P_{n+1}^{(k)} - \left(1 - p_n^{(k-1)}\right) P_n^{(k-1)}\right) / P_n^{(k)}. \tag{12}$$

Using (11) and (12), $p_n^{(k)}$ may be computed for all $n = 0, \ldots, N-1$ and $k = 1, \ldots, n+1$. This constructs a binomial lattice matching the stock price distribution.

We may now construct a multinomial lattice based on the binomial lattice as follows: Consider a two step binomial lattice, where we suppose that the up and down rates, u and d, and probabilities p, $p_1^{(1)}$, and $p_1^{(2)}$ are specified as shown in the left side of Fig. 1. The right side of Fig. 1 is a trinomial tree, where the up, middle, and down states are given by Su^2, Sud and Sd^2. If the up, middle, and down probabilities of the tree are given by

$$p_u = p \cdot p_1^{(1)},\ p_m = p \cdot (1 - p_1^{(1)}) + (1-p) \cdot p_1^{(1)},\ p_d = (1-p) \cdot (1 - p_1^{(1)})$$

then the binomial lattice and the trinomial tree will define the same random walk as far as the initial state and final distributions are concerned, i.e., both random walks have final distributions with identical statistical properties. More generally, in a similar manner one may construct a multinomial lattice with L branches at each node based on a multi-step binomial lattice.

3 Synthesis Methods

Once we have constructed a multinomial lattice, we may apply several techniques for pricing and hedging derivatives. In this section, we demonstrate some of these

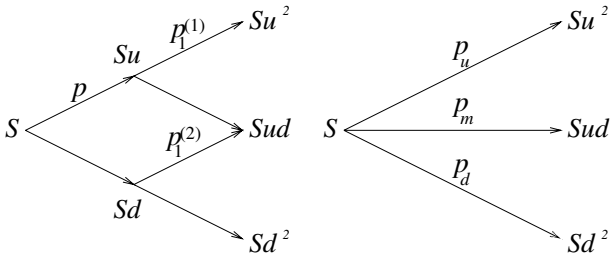


Fig. 1. Trinomial lattice construction

techniques which are used with multinomial lattices. Since most of the following ideas have been considered extensively in the literature, we merely provide a brief outline of them in this paper.

3.1 Mean Square Optimal Hedges

Mean square optimal hedging is a trading strategy which constructs a portfolio whose payoff approximates that of a derivative security as closely as possible in the mean square error sense. Although we only deal with the case of a European call option in this subsection, the same approach can be extended to other types of options, including many exotics (such as barriers, compounds, and others) and options with time optionality (such as Americans and Bermudans).

Let B_n denote the price of a (risk free) bond under the time dependent interest rate r_n where B_n satisfies

$$B_n = (1 + r_n)B_{n-1}, \quad n = 1, \dots, N. \tag{13}$$

at discrete times $t_n = n\tau,\ n = 0,\ 1, \dots, N$. Also, let C_n, $n = 0, 1, \dots, N$ denote the value of a call option with strike price K, which pays

$$C_N = (S_N - K)^+$$

at maturity $t = T$. Finally, we define a portfolio $(\delta_n, \theta_n) \in \Re^2$ indexed by time $n = 0 \dots N$, and let

$$\Omega_n := \delta_n S_n + \theta_n B_n, \quad n = 0 \dots N \tag{14}$$

be the value of the portfolio, where δ_n represents the number of shares of stock and θ_n the number of bonds held by the trader during the time interval $t \in [t_n,\ t_{n+1})$. Finally, we assume that the portfolio is self-financing;

$$\delta_{n-1} S_n + \theta_{n-1} B_n = \delta_n S_n + \theta_n B_n, \quad {}^\forall n = 1 \dots N. \tag{15}$$

We now introduce an optimal hedging strategy to minimize the mean square of the difference between the final payoff of the call option and the value of the

portfolio (i.e. $C_N - \Omega_N$), namely mean square optimal hedging (MSOH):

$$\textbf{MSOH} \left| \begin{array}{l} \text{Minimize}: \ \mathbb{E}\left[(C_N - \Omega_N)^2 \middle| S_0, \ \Omega_0 \right] \\ \text{Subject to}: \ \delta_n \in \Re, \ n = 0, \ldots, N-1, \ \Omega_0 \in \Re \end{array} \right. \tag{16}$$

To obtain the optimal hedging strategy $\delta_k \in \Re, \ k = 0, \ldots, N-1$ and initial portfolio wealth Ω_0, dynamic programming (see e.g., [1]) may be applied once probabilities for possible outcomes for the stock have been assigned. Note that the MSOH problem can be solved very efficiently by dynamic programming if the stock process is modeled on a lattice [7].

3.2 Volatility Smile and Implied Kurtosis

We next discuss pricing models which take the "volatility smile" into account. There is a large body of literature which provides option pricing formulas for smiley options by using binomial lattices, trinomial lattices, or finite difference methods (see e.g., [6,8,14,16,17] and references therein). A common approach is to use market option data to determine a corresponding local volatility function or risk neutral probability distribution to match the volatility smile.

Another approach to modeling the local volatility function for smiley options is to take advantage of the so-called implied kurtosis [3,15]. This approach simply requires an estimate of implied kurtosis which can be extracted from the market price of options. The relation between implied kurtosis and the volatility smile is given by the following equation [3,15]:

$$\sigma(S_n, \ n) = \sigma \left[1 + \frac{\kappa \tau}{24} \left(\frac{(K - S_n)^2}{\sigma^2 S_n^2 T} - 1 \right) \right] \tag{17}$$

where σ can be thought as a "true volatility" corresponding to the variance of the stock price distribution at maturity, and κ is the (annualized) kurtosis of the stock price distribution. Therefore, this formulation provides a connection between implied volatility and a constant volatility with kurtosis.

Given the formula in (17), one can apply local volatility based pricing methods such as trinomial lattices or corresponding finite difference methods (see e.g., [8] and references therein). However it might be more suitable to use a multinomial lattice to take kurtosis (or the fourth cumulant) into account. In this case, one can directly construct a multinomial lattice with five branches as in Subsection 2.1, with constant σ and κ, instead of using a trinomial lattice with a local volatility function. A more sophisticated model can be developed by taking into consideration the time dependence of kurtosis, i.e., k_n, which provides a volatility surface curve. One can then apply standard risk neutral valuation techniques for derivatives pricing.

4 Conclusion

In this paper, we provided a parameterization of multinomial lattice random walks which take cumulants into account. In the binomial and trinomial lattice cases, this parameterization reduced to standard formulas. We showed that

higher order cumulants may be taken into account by using multinomial lattices with four or more branches. Finally, we demonstrated two types of synthesis methods which take advantage of multinomial lattices: mean square optimal hedging in incomplete markets and valuation techniques which use implied volatility or kurtosis.

References

1. Bellman, R.: *Dynamic Programming*. Princeton University Press, Princeton, NJ (1957)
2. Black, F., Scholes, M.; The Pricing of Options and Corporate Liabilities. Journal of Political Economy **81** (1973) 637–654
3. Bouchaud, J-P. and M. Potters.: *Theory of Financial Risks*. Cambridge University Press (2000)
4. Cox, J.C., Ross, S.A.: The valuation of options for alternative stochastic processes. Journal of Financial Economics **3** (1976) 145–166
5. Cox, J.C., Ross, S.A., Rubinstein, M.: Option pricing: A simplified approach. Journal of Financial Economics **7** (1979) 229–263
6. Derman, E., Kani, I.: Riding on a Smile. Risk **7** (1994) 18–20
7. Fedotov, S., Mikhailov, S.: Option Pricing for Incomplete Markets via Stochastic Optimization: Transaction Costs, Adaptive Control, and Forecast. Int. J. of Theoretical and Applied Finance **4** No. 1 (2001) 179–195
8. J. Hull, *Options, Futures, and Other Derivative Securities*, 4th edition. Englewood Cliffs: Prentice-Hall, 1999.
9. Hull, J., White, A.: The Pricing of Options on Assets with Stochastic Volatilities. Journal of Finance **42** (1987) 281–300
10. Jarrow, R., Rudd, A.: Approximate option valuation for arbitrary stochastic processes. Journal of Financial Economics **10** (1982) 347–369
11. Jarrow, R., Rudd, A.: *Option Pricing*. McGraw-Hill Professional Book Group (1983)
12. Karandikar, R.L., Rachev, S.T.: A generalized binomial model and option pricing formulae for subordinated stock-price processes. Probability and Mathematical Statics **15** (1995) 427–447
13. Merton, R.C., Option pricing when underlying stock returns are discontinuous. Journal of Financial Economics **3** (1976) 125–144
14. Pirkner, C.D., Weigend, A.S., Zimmermann, H.: Extracting Risk-Neutral Densities from Option Prices Using Mixture Binomial Trees. Proc. of the IEEE/IAFE/INFORMS Conf. on Computational Intelligence for Financial Engineering (1999) 135–158
15. Potters, M., R. Cont and J-P. Bouchaud.: Financial markets as adaptive systems. Europhys. Lett. **41** No. 3 (1998) 239–244
16. Rubinstein, M.: Implied Binomial Trees. Journal of Finance **3** (1994) 771–818
17. Rubinstein, M.: On the Relation Between Binomial and Trinomial Option Pricing Models. Research Program in Finance Working Papers, ♯RPF-292, University of California at Berkeley (2000)

On Parallel Pseudo-Random Number Generation

Chih Jeng Kenneth Tan

School of Computer Science
The Queen's University of Belfast
Belfast BT7 1NN
Northern Ireland
United Kingdom
cjtan@acm.org

Abstract. Parallel computing has been touted as the pinnacle of high performance digital computing by many. However, many problems remain intractable using deterministic algorithms. Randomized algorithms which are, in some cases, less efficient than their deterministic counterpart for smaller problem sizes, can overturn the intractability of various large scale problems. These algorithms, however, require a source of randomness. Pseudo-random number generators were created for many of these purposes. When performing computations on parallel machines, an additional criterion for randomized algorithms to be worthwhile is the availability of a parallel pseudo-random number generator. This paper presents an efficient algorithm for parallel pseudo-random number generation.

Keywords: Randomized computations, Monte Carlo method, Stochastic methods, Pseudo-random number generators, Parallel computing

1 Introduction

Parallel computing has been touted as the pinnacle of high performance digital computing by many. However, many problems remain intractable using deterministic algorithms even on large parallel digital machines. Randomized algorithms, which are in some cases less efficient than their deterministic counterpart, especially when the problem sizes are relatively small, can overturn the intractability of various large scale problems. In the area of computational finance, for example, stochastic algorithms have been crucial for the solution of various problems. Similar examples may be drawn from areas ranging from computational linear algebra, to computational physics, to environmental modeling.

At the heart of the stochastic algorithms, lie a source of randomness. The question of what can be considered random has often been asked. Various physical sources of randomness have been suggested as sources of randomness for randomized algorithms. Not only that such sources are often not repeatable, making it very hard to verify or debug a program written, it has been shown that some of these physical sources are not "sufficiently random". Pseudo-random number generators are often designed to be a source of randomness which can be used in stochastic computations.

V.N. Alexandrov et al. (Eds.): ICCS 2001, LNCS 2073, pp. 589–596, 2001.
© Springer-Verlag Berlin Heidelberg 2001

This paper presents an efficient algorithm for parallel pseudo-random number generator called `PLFG`, which is based on lagged Fibonacci generator algorithm.

2 Pseudo-Random Number Generators

Pseudo-random number generators have been an interest of researchers, since the early days of computing. Putting aside the philosophical issues involved in the question of what is, or can be, considered random, pseudo-random number generators have to cater for repeatable simulations, have relatively small storage space requirements, and have good randomness properties within the sequence generated.

When performing computations on parallel machines, an additional criterion that needs to be satisfied is the availability of a parallel pseudo-random number generator. The streams of pseudo-random numbers used by each processor have to be independent.

In addition, computational requirements such as coding, initialization time, running time, memory footprint, portability and efficiency have to be taken into consideration as well, when designing pseudo-random number generating algorithms [7].

The pseudo-random number generators used today generally fall into the following categories: linear congruential generators, non-linear congruential generators, lagged Fibonacci generators, Tausworthe generators and mixed generators. Regardless of which pseudo-random number generator is used, its algorithm inflates an input of a short number of bits into a much longer sequence of random bits.

2.1 Lagged Fibonacci Generators

Generally, lagged Fibonacci generators are of the form

$$x_i = (x_{i-p_1} \odot x_{i-p_2}) \bmod M$$

where x_i is the next output pseudo-random number, and $\odot$ is a binary operation. The lag values are p_1 and $p_2, p_1 > p_2$. The operations addition (or subtraction), multiplication or bitwise exclusive OR (XOR) are commonly used in place of $\odot$. M is typically a large integer value or 1 if x_i is a floating point number. When XOR operation is used, mod M is dropped. It is obvious that LFGs require a lag table of length p_1 to store $x_j, j = i-1, i-2, \ldots, i-p_1$.

Although multiple lagged Fibonacci generators using XOR operations can be combined to provide good quality sequences [4], the individual sequence obtained by using a lagged Fibonacci generator with XOR operations give the worst pseudo-random numbers, in terms of their randomness properties [5,1,11]. Multiplicative lagged Fibonacci generators have been shown to have superior properties compared to additive lagged Fibonacci generators [5]. Because multiplication operations are still being perceived as being slower than addition or

subtraction operations, additive lagged Fibonacci generators have found more common use than their multiplicative counterpart. However, tests comparing operation execution times have shown that, with current processors and compilers, multiplication, addition and subtraction operations are of similar speeds. Thus, the argument favoring additive operations over multiplicative operations is nulled and multiplicative lagged Fibonacci generators should be preferred.

Care should be taken when choosing the parameters p_1, p_2 and M in order to obtain a long period and good randomness properties. The value of $p_1 > 1279$ was suggested in [2]. Having a large p_1 also improves randomness since smaller lags lead to higher correlation between numbers in the sequence [5,1,2]. In lagged Fibonacci generators, the key purpose of M is to ensure that the output is bounded within the range of the data type.

Initializing the lag table of the lagged Fibonacci generator is also of critical importance. The initial values have to be independent. To obtain these values, another pseudo-random number generator is often used.

With $M = 2^b$, where b is the total number of bits in the data type, additive lagged Fibonacci generator have a maximal period $\Pi_{ALFG} = 2^{b-1}\,(2^{p_1} - 1)$. The maximal period of multiplicative lagged Fibonacci generator, however, is shorter than that of additive lagged Fibonacci generator: $\Pi_{MLFG} = 2^{b-3}\,(2^{p_1} - 1)$. It is obvious that this shorter period should not pose a problem for multiplicative lagged Fibonacci generators if p_1 is large.

If $x_{i-p_1}, x_{i-p_2}, \ldots, x_{i-p_n}$ are used to generate x_i, it is said to be an "n-tap LFG". Empirical tests have shown that $n > 2$ may increase the randomness quality of the sequence generated. In an n-tap additive lagged Fibonacci generator, M can be chosen to be the largest prime $< 2^b$ [3]. It can be proven using the theory of finite fields that pseudo-random numbers generated in such a manner will be a good source of random numbers. For a complete discussion, see [3].

3 Parallelizing Schemes

Several pseudo-random number generator parallelizing schemes exist. The well known ones being leap frog, sequence splitting, independent sequences and shuffling leap frog [2,12]. All these techniques, except the method of independent sequences, require arbitrary elements of the sequence to be generated efficiently. While it is technically possible to parallelize lagged Fibonacci generators with techniques like leap frog, sequence splitting and shuffling leap frog, the amount of inter-processor communication that would be required makes it impractical to parallelize lagged Fibonacci generators in such a manner. But lagged Fibonacci generators can be parallelized very easily with the independent streams method, and may be very efficient. Pseudo-random number generator parallelization by independent streams is also a recommended technique [8]. The independent sequences are obtained by having multiple generators running on multiple processors, but seeded independently. It should be stressed that seeding the lag tables have to be done with care, to ensure independence between the individual lag tables.

When a parallel pseudo-random number generator with independent sequences is used for Monte Carlo simulations, it is analogous to running the simulation multiple times, each time with a different pseudo-random number generator. This is highly desirable in Monte Carlo simulations since the variance can be reduced by $\mathcal{O}(\sqrt{n})$ if n independent trials are being carried out.

4 Parallel Implementation

For the reasons of the superiority of lagged Fibonacci generators, and multiplicative lagged Fibonacci generator in particular, as discussed in [9], a multiplicative lagged Fibonacci generator algorithm was used as a basis for the pseudo-random number generator implemented. An independent sequences scheme was used for parallelization.

The length of the bit table of a pseudo-random number generator determines the number of parameters needed for its initialization. In the case of an linear congruential generator, where only one past value is used to produce the next output, the size of the bit table is the size of the data type used, and only one value is needed for seeding [3]. For a lagged Fibonacci generator however, the bit table consists of the bits of all $x_i, i = 1, 2, \ldots, p_1$ in the lag table. This feature of the lagged Fibonacci generator may have a positive influence on the quality of the pseudo-random number sequence generated.

A parallel pseudo-random number generator has to generate sequences which are independent of each other. This translates to independence between all the bit tables in the parallel pseudo-random number generator, at any point in time. In parallelizing a pseudo-random number generator by the independent streams, the initial bits in all the bit tables will have to be independent from each other, since the bits of subsequent generated elements are pushed onto the bit table. As such, when parallelizing an lagged Fibonacci generator using independent sequences, the stringent requirement for independence between the seed elements, $x_i, i = 1, 2, \ldots, p_1$, on each processor cannot be stressed any further.

To initialize the lag tables, most existing lagged Fibonacci generators, sequential and parallel, use an linear congruential generator to fill the elements of the lag tables. However, in `PLFG`, the lag tables were chosen to be initialized by a sequential pseudo-random number generator call Mersenne Twister [6]. The Mersenne Twister used has the Mersenne prime period, $\Pi_{MT19937} = 2^{19937} - 1$, thus known as the MT19937. This generator has passed several tests for randomness, including DIEHARD [6]. In addition, MT19937 has been tested to be very efficient, generating 10^7 pseudo-random numbers in 1.76 seconds on an Intel Pentium Pro 200MHz processor.

The lag values of `PLFG` were chosen to be $p_1 = 23209, p_2 = 9739$, recommended by Knuth in [3]. There is nothing in its design which prohibits the extension of the number of taps, n, to $n > 2$. The memory footprint is kept small, while maintaining high efficiency, by using a round-robin algorithm for lag table access.

Table 1. Results of 2D Ising model Monte Carlo simulation test with Metropolis algorithm; ϵ denotes error, σ denotes standard deviation, Cv denotes specific heat.

Generator	ϵ**energy**	σ**energy**	ϵ**Cv**	σ**Cv**
`PLFG`	0.0005960	0.0085104	0.0193600	0.0448184
SPRNG Multiplicative LFG	0.0091587	0.0269140	0.6682971	0.1442130
SPRNG Additive LFG	0.0188618	0.0193388	0.1587430	0.0911916
SPRNG Combined Multiple Recursive Generator	0.0678726	0.0216546	0.7692472	0.1732936

The output pseudo-random number is of type `unsigned long` in ANSI Standard C, which is typically a 32-bit data type on 32-bit architecture machines, but is a 64-bit data type on some 64-bit architecture machines. This dependence on machine architecture resulting in the variability in the period of the pseudo-random number sequence is indeed a feature since its period will automatically expand from when machine with wider word size becomes available. However, when used for simulations running on heterogeneous workstation clusters, this may be a concern.

The total number of independent streams is limited by the period of MT19937. Since total number of independent streams is $\frac{2^{19937}-1}{23209}$, this limitation is moot in practice. Thus, it is practically scalable to as many processors as needed. With $\Pi_{MT19937}$, Π_{MLFG} and p_1, all large, the probability that the sequences overlapping is minimal.

Both initialization and generation of the pseudo-random numbers can be performed in parallel, without any communication needed. The only time when communication is needed, is to synchronize before shutdown. This coarse-grained parallelism is highly desirable for Monte Carlo applications, which are themselves coarse-grained parallel as well.

5 Quality of Sequences Generated

`PLFG` has been subjected to the test using 2D Ising model Monte Carlo simulation with both the Metropolis and the Wolff algorithm.[1] The results shown in Tables 1 and 2 were obtained using identical test parameters. It is clear that the `PLFG` parallel pseudo-random number generator gives superior results compared to the results obtained by parallel pseudo-random number generators provided in the Scalable Pseudo-random Number Generator (SPRNG) package developed at the National Center for Supercomputing Applications, University of Illinois at Urbana-Champaign.[2]

[1] The source code for the tests were ported from the Scalable Pseudo-random Number Generator (SPRNG) package.

[2] At the time of writing, SPRNG Version 2.0 has just been announced. The version of SPRNG considered here is Version 1.0.

Table 2. Results of 2D Ising model Monte Carlo simulation test with Wolff algorithm; ϵ denotes error, σ denotes standard deviation, Cv denotes specific heat.

Generator	ϵ**energy**	σ**energy**	$^{\epsilon}$**Cv**	$^{\sigma}$**Cv**
`PLFG`	0.0026430	0.0030601	0.0054660	0.0271471
SPRNG Multiplicative LFG	0.0541587	0.0287093	0.5353881	0.1165021
SPRNG Additive LFG	0.0660337	0.0291500	0.4078441	0.1705614
SPRNG Combined Multiple Recursive Generator	0.0130649	0.0105590	0.0444265	0.1105693

`PLFG` has also been put to test, against the multiplicative lagged Fibnacci generator from the SPRNG package, using the Relaxed Monte Carlo method for solving of systems of linear algebraic equations [10]. Comparing the results shown in Tables 3 and 4, it is clear that `PLFG` is at least on par with the multiplicative lagged Fibnacci generator from the SPRNG package, if not a better parallel pseudo-random number generator. The tests were done on a DEC Alpha XP1000 cluster, with EV67 processors running at 667MHz.

Table 3. Relaxed Monte Carlo method with `PLFG`, using 10 processors, on a DEC Alpha XP1000 cluster.

Data set	**Norm**	**Solution time (sec.)**	**RMS error**	**No. chains**
1000-A1	0.5	7.764	1.91422e-02	2274000
1000-A2	0.6	7.973	1.92253e-02	2274000
1000-A3	0.7	7.996	1.93224e-02	2274000
1000-A4	0.8	7.865	1.91973e-02	2274000
1000-A5	0.5	7.743	1.27150e-02	2274000
1000-A6	0.6	7.691	1.27490e-02	2274000
1000-A7	0.7	7.809	1.27353e-02	2274000
1000-A8	0.8	7.701	1.27458e-02	2274000

Timing tests for generating 10^6 pseudo-random numbers per stream have also been conducted. Results for tests conducted on both a cluster of DEC Alpha machines with Alpha 21164 500 MHz processors, connected via Myrinet, and a dual processor Intel x86 machine with Pentium Pro 200 MHz processors are shown in Table 5.[3] For tests on the DEC Alpha cluster, 20 processors were used. It can be seen that the speed of `PLFG` is on par with other parallel pseudo-random number generators.

[3] The Alpha 21164 and Pentium Pro 200 processors both have on-chip instruction and data L1 caches of 8Kb each, and 96Kb and 256 Kb L2 cache respectively.

Table 4. Relaxed Monte Carlo method with SPRNG MLFG, using 10 processors, on a DEC Alpha XP1000 cluster.

Data set	Norm	Solution time (sec.)	RMS error	No. chains
1000-A1	0.5	7.842	4.43195e-02	2274000
1000-A2	0.6	7.842	4.53718e-02	2274000
1000-A3	0.7	8.666	4.78022e-02	2274000
1000-A4	0.8	8.087	4.77088e-02	2274000
1000-A5	0.5	8.138	3.17604e-02	2274000
1000-A6	0.6	7.748	3.17574e-02	2274000
1000-A7	0.7	8.172	3.18349e-02	2274000
1000-A8	0.8	7.392	3.17931e-02	2274000

Table 5. Average time taken for generating 10^6 pseudo-random numbers.

Generator	Average time (sec.)	
	Intel Pentium Pro	DEC Alpha
`PLFG`	0.614	0.251
SPRNG Multiplicative LFG	0.720	0.187
SPRNG 64-bit LCG	1.510	0.078
SPRNG Additive LFG	0.270	0.260
SPRNG Combined Multiple Recursive Generator	2.910	0.238

6 Conclusion

`PLFG` is a highly efficient and scalable parallel pseudo-random number generator. Initialization of the lag tables can be sped up using another highly efficient pseudo-random number generator with good randomness qualities, while yielding quality sequences, as seen in the results of empirical tests conducted. In addition, coarse-grained parallelism employed in parallelizing `PLFG` and its scalability makes it extremely suitable for Monte Carlo simulations.

7 Acknowledgment

The author would like to thank J. A. Rod Blais from the Pacific Institute for the Mathematical Sciences, and Christiane Lemieux from the Department of Mathematics and Statistics, both at the University of Calgary, Canada, M. Isabel Casas Villalba from Norkom Technologies, Ireland, and Vassil Alexandrov from the High Performance Computing Center, University of Reading, UK, for their support and the fruitful discussions.

References

[1] Coddington, P. D. Analysis of Random Number Generators Using Monte Carlo Simulation. *International Journal of Modern Physics C5* (1994).

[2] Coddington, P. D. Random Number Generators for Parallel Computers. *National HPCC Software Exchange Review*, 1.1 (1997).

[3] Knuth, D. E. *The Art of Computer Programming, Volume II: Seminumerical Algorithms*, 3 ed. Addison Wesley Longman Higher Education, 1998.

[4] L'Ecuyer, P. Maximally Equidistributed Combined Tausworthe Generators. *Mathematics of Computation 65*, 213 (1996), 203 – 213.

[5] Marsaglia, G. A Current View of Random Number Generators. In *Computing Science and Statistics: Proceedings of the XVI Symposium on the Interface* (1984).

[6] Matsumoto, M., and Nishimura, T. Mersenne Twister: A 623-Dimensionally Equidistributed Uniform Pseudo-Random Number Generator. *ACM Transactions on Modeling and Computer Simulation 8*, 1 (1998), 3 – 30.

[7] Niederreiter, H. *Random Number Generation and Quasi-Monte Carlo Methods*. No. 63 in CMBS-NSF Regional Conference Series in Applied Mathematics. Society for Industrial and Applied Mathematics, 1992.

[8] Srinivasan, A., Ceperley, D., and Mascagni, M. Testing Parallel Random Number Generators. In *Proceedings of the Third International Conference on Monte Carlo and Quasi-Monte Carlo Methods in Scientific Computing* (1998).

[9] Tan, C. J. K. Efficient Parallel Pseudo-random Number Generation. In *Proceedings of the 2000 International Conference on Parallel and Distributed Processing Techniques and Applications* (2000), H. R. Arabnia, et al., Ed., vol. 1, CSREA Press.

[10] Tan, C. J. K., and Alexandrov, V. N. Relaxed Monte Carlo Method for Solution of Systems of Linear Algebraic Equations. In *Recent Advances in Computational Science* (2001), V. N. Alexandrov, J. J. Dongarra, and C. J. K. Tan, Eds., vol. 2073 of *Lecture Notes in Computer Science*, Springer-Verlag.

[11] Vattulainen, I., Ala-Nissila, T., and Kankaala, K. Physical Models as Tests of Randomness. *Physics Review E52* (1995).

[12] Williams, K. P., and Williams, S. A. Implementation of an Efficient and Powerful Parallel Pseudo-random Number Generator. In *Proceedings of the Second European PVM Users' Group Meeting* (1995).

A General Framework for Trinomial Trees

Ali Lari-Lavassani and Bradley D. Tifenbach

Mathematical and Computational Finance Laboratory. Department of Mathematics and Statistics. University of Calgary. Calgary, Alberta T2N 1N4

Abstract. Three general trinomial option pricing methods are formally developed and numerically implemented and explored. Applications to American option pricing are presented for one and two factor models.

1 Introduction

Hull and White introduced trinomial trees for processes with additive noise and linear drift. In this work we unify the abstract features of these constructions and generalize them to encompass the case of nonlinear drifts, and outline some general *Conditions* such constructions should satisfy. Increasing computing performance allows for actual implementations of these methods in trading environments. Since our ultimate objective is to develop different algorithms, we assume throughout, that all processes are in a risk neutral world, see to [T, 00] for more on these issues, and [JW, 00] for many up to date references.

2 Continuous Processes

2.1 Generalities

Consider the following stochastic differential equation (SDE)

$$ds_t = a(s_t, \theta(t))\ dt + b\,(s_t)\ \ dz_t \tag{1}$$

where the drift and volatility functions a and b satisfy the usual integrability conditions described, e.g., in [KP, 99] and the parameter $\theta(t)$ is a continuous function of time designed to capture a given term structure or the seasonal shape of the expectation curve $\varphi(t) = E(s_t \mid s_0)$ for $t \in [0, T]$. The construction of additive trinomial trees requires constant standard deviations. We henceforth assume that the following transformation exists and is invertible, leading to the new variables

$$S = \sigma \int \frac{ds}{b(s)} \quad , \quad S_t := S(s_t) \quad , \quad s_t := s(S_t).$$

Then by the Ito formula we have

$$dS_t = A(S_t, \theta(t))\ dt + \sigma\ dz_t\ , \ \text{ with } \quad A(S_t, \theta(t)) := \sigma\left(\frac{a(s_t, \theta(t))}{b(s_t)} - \frac{b'(s_t)}{2}\right). \tag{2}$$

We next discuss mean reverting processes since they will be used as examples.

V.N. Alexandrov et al. (Eds.): ICCS 2001, LNCS 2073, pp. 597–606, 2001.
© Springer-Verlag Berlin Heidelberg 2001

2.2 Mean Reverting Processes with Additive Noise

[HW, 94 a,b] develop models with additive noise, suitable for short term interest rates. In a slightly modified notation their one factor model writes as

$$ds_t = \alpha(l(t) - s_t)\ dt + \sigma\ dz_t \tag{3}$$

where α and σ are constant, $l(t)$ is the time varying reversion level.

Their two factor model is the system

$$ds_t = \alpha(l(t) + v_t - s_t)\ dt + \sigma_1\ dz_t^1 \quad , \qquad dv_t = -\delta\ v_t\ dt + \sigma_2\ dz_t^2 \tag{4}$$

where $v_0 = 0$, the parameters α, δ, σ_1 and σ_2 are constants and the Brownian motions have instantaneous correlation ρ_{12}. Assuming the generic condition, $\alpha \neq \delta$, this system decouples via the new variable $y_t = s_t + v_t/(\delta - \alpha)$:

$$dy_t = \alpha(l\,(t) - y_t)\ dt + \sigma_3\ dz_t^3 \quad , dv_t = -\delta\ v_t\ dt + \sigma_2\ dz_t^2 \tag{5}$$

where $\sigma_3^2 = (\sigma_1^2(\delta - \alpha)^2 + 2\ \rho_{12}\ \sigma_1\ \sigma_2(\delta - \alpha) + \sigma_2^2\ )/(\delta - \alpha)^2$and z_t^3 is another Brownian motion, with the correlation between z_t^2 and z_t^3 being given by $\rho_{23} = (\rho_{12}\sigma_1 + \sigma_2/(\delta - \alpha))/\sigma_3$.

2.3 Mean Reverting Processes with Multiplicative Noise

[P, 98] introduces processes with multiplicative noise and constant coefficients to model energy spot prices. A partial study of the dynamics of these equations and implementations via binomial trees, can be found in [LSW, 00]. For generalizations of these models and numerical implementations see [T, 00]. We follow the latter and allow one of the parameters, see $l(t)$ below, to be a function of time, in order to capture seasonality or match the term structure of forward markets. The generalized one factor mean reverting model with multiplicative noise is

$$ds_t = \alpha(l\,(t) - s_t)dt + \sigma\ s_t\ dz_t \tag{6}$$

where α and σ are constant and $l(t)$ is the time varying reversion level. We next transform this equation into an additive process by putting $S_t = \ln s_t$. Then the Ito formula yields (after also substituting $L(t) = \ln l(t)$)

$$dS_t = (\alpha(e^{L(t)-S_t} - 1) - \frac{\sigma^2}{2})dt + \sigma\ dz_t. \tag{7}$$

Note that the drift is no longer linear. The generalized two factor system is

$$ds_t = \alpha\,(l_t - s_t)\,dt + \sigma_1\ s_t\ dz_t^1, \quad dl_t = \beta\,(t)\ \ l_t\ dt + \sigma_2\ l_t\ dz_t^2 \tag{8}$$

where the parameters α, σ_1 and σ_2 are constants, $\beta\,(t)$ captures the term structure and or seasonality of forward markets, and z_t^1 and z_t^2 are Brownian motions with instantaneous correlation ρ_{12}. Under the change of variable $S_t = \ln s_t$ and $L_t = \ln l_t$, the system becomes

$$dS_t = (\alpha(e^{L_t - S_t} - 1) - \frac{\sigma^2}{2})\ dt + \sigma_1\ dz_t^1,\ \ dL_t = (\beta\,(t) - \frac{\sigma_2^2}{2})\ dt + \sigma_2\ dz_t^2.$$

To decouple this system introduce the variable $Y_t = L_t - S_t$ so that

$$dY_t = \alpha\ (B(t) - e^{Y_t})\ dt + \sigma_3\ dz_t^3 \quad , \quad dL_t = (\beta\,(t) - \frac{\sigma_2^2}{2})\ dt + \sigma_2\ dz_t^2 \qquad (9)$$

where, $B(t) = 1 + \frac{1}{\alpha}(\beta(t) + \frac{\sigma_1^2 - \sigma_2^2}{2})$, $\sigma_3^2 = \sigma_1^2 - 2\rho_{12}\ \sigma_1\sigma_2 + \sigma_2^2$ and z_t^3 is another Brownian motion, with the correlation between z_t^2 and z_t^3 being $\rho_{23} = (\sigma_2 - \rho_{12}\ \sigma_1)/\sigma_3$. Note that (9) is in the format required for trinomial tree construction.

3 Trinomial Trees

3.1 Infinitesimal Structure

For the SDE (2), denote the mean and variance of the displacement $\Delta S_t = S_{t+\Delta t} - S_t$ by $M_t(\Delta t)$ and $V_t(\Delta t)$ respectively. We then have the expansion

Proposition 1. *$M_t(\Delta t) = A(S_t, \theta(t))\Delta t + O(\Delta t^2)$ and $V_t(\Delta t) = \sigma^2\Delta t + O(\Delta t^2)$.*

Proof. $M_t(\Delta t) = \int_t^{t+\Delta t} E(A(S_u, \theta_u)|A(S_t, \theta(t))\ du$. Expanding the integrand yields, $M_t(\Delta t) = \int_t^{t+\Delta t}(A(S_t, \theta(t)) + O(\Delta t))\ du$ and hence the result. Now, $V_t(\Delta t) = E[(S_{t+\Delta t} - M_t(\Delta t) - S_t)^2] = E[(\int_t^{t+\Delta t} A(S_u, \theta(u))\ du + \int_t^{t+\Delta t} \sigma\ du - M_t(\Delta t))^2] = E[(\int_t^{t+\Delta t} \sigma\ du)^2 + O(\Delta t^2)]$. After using a theorem in [KP, 99] p. 86, the latter becomes, $(\int_t^{t+\Delta t} E(\sigma^2)\ du) + O(\Delta t^2) = \sigma^2\Delta t + O(\Delta t^2)$.

3.2 The Discrete Process

Discretize the interval $[0, T]$ into n time steps of length $\Delta t = T/n$, set $t_i = i\ \Delta t$ and let $S_{t_i} = S_{ij}$. A trinomial tree for S_t is a discrete process on a two dimensional lattice whose integer nodes are indexed by (i, j). From (i, j), over the interval $[t_i, t_{i+\Delta t}]$, it is only possible to branch to one of the three nodes $(i + 1, h_{ij} + 1)$, $(i + 1, h_{ij})$ or $(i + 1, h_{ij} - 1)$, called respectively, the *up*, *middle* and *down* nodes, with respective probabilities $p_{ij}^{(u)}$, $p_{ij}^{(m)}$ and $p_{ij}^{(d)}$. By definition, h_{ij} is assigned so that $S_{i+1,h_{ij}}$ is as close as possible to the expected value $E(S_{t_i+\Delta t}|S_{t_i} = S_{ij})$. To remove extra degrees of freedom, we suppose that the up and down jumps have increments of equal length from the middle node:

Condition 1. $\Delta S_{ij} := S_{i+1,h_{ij}+1} - S_{i+1,h_{ij}} = S_{i+1,h_{ij}} - S_{i+1,h_{ij}-1}$.

Let $\eta_{t_i}(\Delta t) = E(S_{t_i+\Delta t}|S_{t_i} = S_{ij}) - S_{i+1,h_{ij}}$ be the offset between the expected value and the middle node. Since by definition, $M_{t_i}(\Delta t) = E(S_{t_i+\Delta t}|S_{t_i} = S_{ij}) - S_{ij}$ we also have $\eta_{t_i}(\Delta t) = S_{ij} + M_{t_i}(\Delta t) - S_{i+1,h_{ij}}$. Now by the very definition of h_{ij} it follows:

Lemma 1. *With the above notation,* $\eta_{t_i}(\Delta t) < \Delta S_{ij}/2$.

Note that $S_{ij} = S_{i0} + j\ \Delta S_{ij}$, where S_{i0}, the position of the *median node* of the i^{th} branch, and the analytical form of h_{ij} will be defined for each of the tree constructions developed next; in all cases $S_{00} = S_0$. This construction allows for multiple jumps. The maximum and minimum values of j are recursively defined by setting $j_{\max}(0) = j_{\min}(0) = 0$, and for $i = 1, ..., n$, $j_{\max}(i) = h_{i-1,j_{\max}(i-1)} + 1$ and $j_{\min}(i) = h_{i-1,j_{\min}(i-1)} - 1$. This relies on the natural

Condition 2. $h_{ij} < h_{ij'}$ for $j < j'$.

By definition of h_{ij} this is the case if $E(S_{t_i+\Delta t}|S_{t_i} = S_{ij}) < E(S_{t_i+\Delta t}|S_{t_i} = S_{ij'})$. This is equivalent to $S_t + M_t(\Delta t)$ being increasing in S_t, and leads to:

Proposition 2. *Suppose* $1 + \frac{d}{dS_t} M_t(\Delta t) > 0$, *then Condition 2 holds.*

Remark 1. In practice it is enough to satisfy the above hypothesis to the order $O(\Delta t)$ and for Δt small enough.

Lemma 2. *For the processes (3), (7) and (9) the hypothesis of the above Proposition holds if* Δt *is chosen small enough.*

Proof. We use Proposition 2. The linear case (3) is trivial, as for (7) and (9), let L denote $l(t)$ or L_t. Then $1 + \frac{d}{dS} M_t(\Delta t) = 1 - \alpha\ e^{L-S_t}\ \Delta t$. By mean reversion $L - S_t$ cannot grow large and since the time horizon $[0, T]$ is compact, $L - S_t$ is bounded. Hence Δt can be chosen small enough to yield the result.

Matching the first and second moments of the continuous processes (2) and the above discrete process over every subinterval $[t_i, t_{i+\Delta t}]$ leads to the system

$$\begin{aligned} p_{ij}^{(u)}(S_{i+1,h_{ij}+1} - S_{ij}) + p_{ij}^{(m)}(S_{i+1,h_{ij}} - S_{ij}) + p_{ij}^{(d)}(S_{i+1,h_{ij}-1} - S_{ij}) &= M_{t_i}(\Delta t) \\ p_{ij}^{(u)}(\Delta S_{ij} - \eta_{t_i}(\Delta t))^2 + p_{ij}^{(m)}\eta_{t_i}^2(\Delta t) + p_{ij}^{(d)}(\Delta S_{ij} + \eta_{t_i}(\Delta t))^2 &= V_{t_i}(\Delta t) \\ p_{ij}^{(u)} + p_{ij}^{(m)} + p_{ij}^{(d)} &= 1 \end{aligned}$$

which has for solutions

$$p_{ij}^{(u)} = \frac{1}{2}\left(\frac{V_{t_i}(\Delta t) + \eta_{t_i}^2(\Delta t)}{\Delta S_{ij}^2} + \frac{\eta_{t_i}(\Delta t)}{\Delta S_{ij}}\right) \quad , \quad p_{ij}^{(m)} = 1 - \frac{V_{t_i}(\Delta t) + \eta_{t_i}^2(\Delta t)}{\Delta S_{ij}^2}$$

$$p_{ij}^{(d)} = \frac{1}{2}\left(\frac{V_{t_i}(\Delta t) + \eta_{t_i}^2(\Delta t)}{\Delta S_{ij}^2} - \frac{\eta_{t_i}(\Delta t)}{\Delta S_{ij}}\right)$$

To remove one degree of freedom we now make the assumption

Condition 3. $\Delta S_{ij} = \sqrt{3V_{ij}(\Delta t)}$.

Note that [HW, 90] suggests this assumption in the infinitesimal limit as $\Delta t \to 0$. Using Condition 3 in the above equations yields the following formulas generalizing those of [HW, 94 a,b], after dropping the Δt in $\eta_t(\Delta t)$:

$$p_{ij}^{(u)} = \frac{1}{6} + \frac{1}{2}\left(\frac{\eta_{t_i}^2}{\Delta S_{ij}^2} + \frac{\eta_{t_i}}{\Delta S_{ij}}\right),\ p_{ij}^{(m)} = \frac{2}{3} - \frac{\eta_{t_i}^2}{\Delta S_{ij}^2},\ p_{ij}^{(d)} = \frac{1}{6} + \frac{1}{2}\left(\frac{\eta_{t_i}^2}{\Delta S_{ij}^2} - \frac{\eta_{t_i}}{\Delta S_{ij}}\right). \tag{10}$$

These probabilities are in $[0,1]$. Indeed both $p_{ij}^{(u)}$ and $p_{ij}^{(d)}$ can be viewed as quadratic expressions of $\eta_{t_i}(\Delta t)/\Delta S_{ij}$ with negative discriminants, leading to positive values. It then suffices to verify that $p_{ij}^{(u)} + p_{ij}^{(d)} \leq 1$ and this follows from Lemma 1. The above can be summarized in

Theorem 1. *Assuming conditions 1, 2 and 3, and matching the first and second moments $M_t(\Delta t)$, $V_t(\Delta t)$ of the continuous process with those of the discrete trinomial process, at each node (i,j), lead to a trinomial tree whose probabilities are given by (10). Furthermore, all probabilities $p_{ij}^{(u)}$, $p_{ij}^{(m)}$ and $p_{ij}^{(d)}$ are in $[0,1]$.*

Remark 2. The complete tree specification still requires to determine h_{ij}. This will depend on the tree geometry adopted and the actual SDE considered.

Remark 3. Condition 3 and Proposition 1 yield the values $\Delta S_{ij} = \sigma\sqrt{3\Delta t} + O(\Delta t)$ and $M_t(\Delta t) = A(S_t, \theta(t))\Delta t + O(\Delta t^2)$. Therefore once h_{ij} is known the entire tree is known.

Remark 4. This trinomial tree is Z_2-symmetric. Indeed, let $Z_2 = \{-1, 1\}$ act on $\{u, m, d\}$ by: $-1.u = d$, $-1.d = u$, $1.m = m$. This action holds both for the nodes and the probabilities.

4 Three Tree Geometries

4.1 Fixed Grid Geometry (FGG)

In FGG the nodes are arranged in a fixed rectangular grid. All positions are referenced relative to the root. That is $S_{i0} = S_0$ for all i, and for $j \in [j_{\min}(i), j_{\max}(i)]$

$$S_{ij} = S_0 + j\Delta S_{ij}, h_{ij} = \left[j + \frac{M_{t_i}(\Delta t)}{\Delta S_{ij}}\right],\ \eta_{t_i}(\Delta t) = M_{t_i}(\Delta t) - (h_{ij} - j)\Delta S_{ij},$$

where here and in the sequel, [ ] denotes the nearest integer.

4.2 Drift Adapted Geometry (DAG)

In DAG one first defines the median nodes Ψ_i as being precisely connected by the drift of the process. Each branch of the tree is then shifted up or down from these median nodes. That is for $j \in [j_{\min}(i), j_{\max}(i)]$, the tree is specified by

$$\Psi_0 = S_0\ \ ,\ \ m_i(\Delta t) = E(S_{t_i+\Delta t} | S_{t_i} = \Psi_i) - \Psi_i, \quad \Psi_i = S_0 + \sum_{k=0}^{i-1} m_k(\Delta t)$$

$$S_{ij} = \Psi_i + j\Delta S_{ij}\ ,\ \ h_{ij} = \left[j + \frac{M_{t_i}(\Delta t) - m_i(\Delta t)}{\Delta S_{ij}}\right].$$

Note that $\eta_{t_i}(\Delta t) = M_{t_i}(\Delta t) - m_i(\Delta t) - (h_{ij} - j)\Delta S_{ij}$, and by construction, those associated with all median nodes $(i, 0)$ are all zero; consequently, by (10), the branching probabilities of all median nodes are $p^{(u)}_{i0} = 1/6$, $p^{(m)}_{i0} = 2/3$, and $p^{(d)}_{i0} = 1/6$. Finally, note that $m_i(\Delta t) = A(\Psi_i, \theta(t_i))\Delta t + O(\Delta t^2)$.

4.3 Forward Tree Geometry (FTG)

Forward Trees are constructed in two stages. We first construct a preliminary tree and then shift its median nodes $\widehat{S}_{i0}$ onto the expected values $\Phi(t_i) = \mathrm{E}(S_{t_i} \mid S_0)$, for all i. We call the SDE (2) *preliminarizable* if for some constant $\widehat{\theta}$

$$A(0, \widehat{\theta}) = 0 \text{ and } \frac{\partial A}{\partial \theta}(0, \widehat{\theta}) \neq 0. \tag{11}$$

Then by the implicit function theorem, there is a unique curve $\theta(S)$ defined for (S, θ) near $(0, \widehat{\theta})$ so that $A(\theta(S), \theta) = 0$. We next define the *preliminarization* of S_t to be the process $\widehat{S}_t$ defined by

$$d\widehat{S}_t = A(\widehat{S}_t, \widehat{\theta})dt + \sigma\ dz \quad \text{with} \quad \widehat{S}_0 = 0.$$

Condition 4. $\widehat{\Phi}(t) := \mathrm{E}(\widehat{S}_t \mid \widehat{S}_0 = 0) = 0 + O(\Delta t^2)$ for all $t \in [0, T]$.

Heuristically (11) yields Condition 4, indeed by Proposition 1, $\mathrm{E}(\widehat{S}_{t+\Delta t} | \widehat{S}_t) - \widehat{S}_t = A(\widehat{S}_t, \widehat{\theta})\ \Delta t + O(\Delta t^2)$, starting at $t = 0$, one would get, by (11) $\widehat{\Phi}(\Delta t) = 0 + O(\Delta t^2)$ and continuing in this manner n times, leads to a total error of $nO(\Delta t^2) = O(\Delta t)$. The *preliminary tree* is then the trinomial tree for $\widehat{S}_t$, constructed using either FGG or DAG. For $j \in [j_{\min}(i), j_{\max}(i)]$, $\widehat{S}_t$ at node (i, j) is given by

$$\widehat{S}_j = j\ \Delta S_{ij}\ ,\ \ \widehat{h}_{ij} = \left[j + \frac{\widehat{M}_{t_i}(\Delta t)}{\Delta S_{ij}} \right],\ \widehat{\eta}_{t_i} = \widehat{M}_{t_i}(\Delta t) - (\widehat{h}_{ij} - j)\Delta S_{ij}$$

Note that the above data do not depend on i hence one needs only to compute $\{\max j_{\max}(i), i \in [0, n]\} - \{\min j_{\min}(i),\ i \in [0, n]\} + 1$ sets of node data. The final tree is formed by shifting the median nodes of the preliminary tree $\widehat{S}_j$ onto $\Phi(t_i)$, while maintaining branching probabilities: the node (i, j) in the final Forward Tree for S_t is $S_{ij} = \Phi(t_i) + \widehat{S}_j$.

We now address the important issue of the validity of the FTG construction, which we distinguish by a hat superscript. The DAG and FTG are approximations of (2), if they are obtained by matching the first and second moments of this SDE. This implies that $\eta_{t_i}(\Delta t)$ should yield the same values, to the order (Δt), for both trees. Hence, $\widehat{M}_{t_i}(\Delta t) - (\widehat{h}_{ij} - j)\Delta S_{ij} = M_{t_i}(\Delta t) - m_i(\Delta t) - (h_{ij} - j)\Delta S_{ij}$. Assuming that almost everywhere on these trees $\widehat{h}_{ij} = h_{ij}$, we then have

Proposition 3. *With the above notations, the DAG and FTG trees yield the same option values if* $\widehat{M}_{t_i}(\Delta t) = M_{t_i}(\Delta t) - m_i(\Delta t)$*; or up to* $O(\Delta t)$*,*

$$A(j\Delta S_{ij}, \widehat{\theta}) = A(\Psi_i + j\Delta S_{ij}, \theta(t_i)) - A(\Psi_i, \theta(t_i)) = \sum_{k=1}^{\infty} \frac{\partial^k A}{\partial S^k}(\Psi_i, \theta(t_i)) \frac{(j\Delta S_{ij})^k}{k!}. \tag{12}$$

Proposition 4. *The process (3) is preliminarizable and satisfies (12). The same is true for (7) and (9) if* $\sigma \ll \alpha$ *and provided mean reversion is strong.*

Proof. In the linear case (3), $\theta(t) = l(t)$ and $A(s_t, l(t)) = \alpha(l(t) - s_t)$. Then (11) obviously holds, and (12) reduces to the true identity $\alpha(0 - j\Delta s_{ij}) = -\alpha j \Delta s_{ij}$. As for (7), $\theta(t) = L(t)$ and $A(S_t, L(t)) = \alpha(e^{L(t) - S_t} - 1) - \sigma^2/2$. Then (11) holds with $\widehat{\theta} = \ln(2\alpha + \sigma^2)/2\alpha$ and (12) leads to $\frac{2\alpha+\sigma^2}{2\alpha}\, \alpha(\, e^{-j\Delta S_{ij}} - 1) = \alpha\, e^{L(t) - \Psi_t}(e^{-j\Delta S_{ij}} - 1)$. If $\sigma \ll \alpha$, then $2\alpha + \sigma^2/2\alpha \approx 1$; also strong mean reversion forces $L(t) - \Psi_i \approx 0$, and hence the result follows. Regarding (9) the argument is analogous for S_t and it is trivially true for L_t.

Remark 5. The above propositions provide a rigorous justification for the famous tree construction of Hull and White. It also establishes that the construction can be used in the nonlinear case but some errors might be expected.

The main difficulty in implementing FTG is to compute $\Phi(t)$ while matching forward market features and Term Structures. If the drift of (1) has an affine functional form, say $a(s_t, \theta(t) = f(t)\ s_t + g(t)$, then the expected value $\varphi(t) = \mathrm{E}(s_t \mid s_0)$ satisfies the ordinary differential equation $\dot{\varphi}(t) = a(\varphi(t), \theta(t))$. Then given the parameter $\theta(t)$ in a functional form exogenously or as a vector matching forward market data, it is always possible to solve for $\varphi(t)$. It is however not true that $\Phi(t) = S(\varphi(t))$. One can still manage to calculate the transformed expectations, by ensuring that they are consistent with the expected value equations $\varphi_i = \sum_{j=j_{\min}(i)}^{j_{\max}(i)} P_{ij} s_{ij}$ at every branch in the tree, where P_{ij} is the probability of reaching node (i, j). Provided we have calculated the branching probabilities at all nodes by (10), the P_{ij}'s may be computed recursively by $P_{00} = 1$ and

$$P_{ij} = \sum_k P_{i-1,k} q[(i-1, k) \to (i, j)]$$

for $i \in [1, n]$, where $q[(i-1, k) \to (i, j)]$ is the probability of branching from node $(i-1, k)$ to node (i, j). Since at node (i, j), s_t is given by the inverse transformation $s_{ij} = s(S_{ij})$, the desired Φ_i's are defined implicitly, for $i \in [0, n]$, by the following equations, which can always be solved by an iterative technique,

$$\varphi_i = \sum_{j=j_{\min}(i)}^{j_{\max}(i)} P_{ij} s(\Phi_i + j\ \Delta S_{ij}) \tag{13}$$

Table 1. American Call Option on a One Factor Model with Additive Noise

Time	FGG			DAG			FTG		
Step	Opt.	Time	Error	Opt.	Time	Error	Opt.	Time	Error
100	277588	0.052	1748	278184	0.052	3671	278247	0.058	4272
200	277643	0.198	1205	278025	0.209	2081	278056	0.103	2354
400	277706	0.791	578	277935	0.834	1189	277951	0.484	1306
800	277743	3.157	207	277868	3.326	519	277876	1.629	558

Remark 6. When the original process s_t has additive noise as in the Hull and White equations, the above procedure can be greatly simplified. Indeed, in this case it is not necessary to transform to another stochastic variable S_t, before building the Forward Tree. In other words, we construct a tree directly for s_t. Therefore, s_t and their preliminarizations $\widehat{s}_t$ are positioned at $\widehat{s}_j = \varphi(t_i) + j\ \Delta s$ and $\widehat{s}_j = j\ \Delta s$, respectively, for $i \in [1, n]$ and $j \in [j_{\min}(i), j_{\max}(i)]$, and most importantly, it is never needed to employ (13). This drastically reduces the computational cost.

5 Numerical Applications to American Options

We now numerically explore the algorithms discussed. To implement two factor models via trinomial trees, we use the standard technique introduced by [HW, 94] consisting in building a tree for each security separately, forming the direct product of the trees and subsequently adjusting the branching probabilities to induce correlation. Implementing nonlinear models are new and have not received much attention in the literature as they are quite harder than the linear cases. For these we choose as underlying process energy spot prices. We price daily American call options. The risk free interest rate is set to be 0.05, time to maturity is 0.25, and we denote by K the strike price. The errors reported are the differences between the option value and the "true" value which is obtained by running each method for high number of time steps n. Our goal is to only demonstrate the convergence patern and the the efficiency of the algorithms.

5.1 Models with Additive Noise

Consider the one factor model (3) with $l(t) = 0.03\ e^{0.1t}$, $\alpha = 3$, $\sigma = 0.015$, $s_0 = 0.03$ and $K = 0.03$. The "true" values are obtained for $n = 1600$. Time is in seconds, option values are to be multiplied by 10^{-10} and the errors by 10^{-11}. The results are reported in Table 1. As for the two factor model (5), $l(t)$ is the same and $\alpha = 3$, $\delta = 0.1$, $\sigma_1 = 0.01$, $\sigma_2 = 0.0145$, $\rho_{12} = 0.6$, $s_0 = 0.03$ and $K = 0.03$. The "true" value is for $n = 400$ and time is in 1000 seconds. Option values are to be multiplied by $10^{-8}$and the errors by 10^{-9}. The results are reported in Table 2.

Table 2. American Call Option on a Two Factor Model with Additive Noise

Time	FGG			DAG			FTG		
Step	Opt.	Time	Error	Opt.	Time	Error	Opt.	Time	Error
50	239966	0.212	3289	239994	0.230	3569	240119	0.198	4667
100	239787	1.748	1501	239789	1.878	1526	239852	1.574	1992
150	239721	3.268	846	239725	3.526	882	239767	2.927	1140
200	239688	7.745	515	239689	8.311	519	239720	6.905	674

Table 3. American Call Option on a One Factor Model with Multiplicative Noise

Time	FGG			DAG			FTG		
Step	Opt.	Time	Error	Opt.	Time	Error	Opt.	Time	Error
100	1.4335	0.9310	60	1.4339	0.8810	63	1.4317	0.721	52
200	1.4303	3.2550	27	1.4304	3.3250	28	1.4289	2.573	24
400	1.4287	12.558	11	1.4288	13.119	12	1.4276	9.784	10
800	1.4280	51.164	4	1.4280	52.646	4	1.4269	39.39	3

5.2 Models with Multiplicative Noise

Let $p(t) = 12.57\, e^{0.80t} - 0.94 \cos 2\pi t + 0.02 \sin 2\pi t$. With 1998 NYMEX spot crude oil data, we imposed in (7) that $l(t)$ models trend and seasonal effects with a general expression involving exponential and periodic functions. This leads after calibration to $l(t) = p(t)$, $\alpha = 36.7$ and $\sigma = 0.336$. We use $S_0 = 12.5$, $K = 13.50$. The "true" value obtained for $n = 1600$, is 1.4276 for FGG and DAG, and 1.4265 for FTG. The unit for computational cost is in seconds and the reported errors are to be multiplied by 10^{-4}. The results are reported in Table 3.

Using techniques such as those discussed in [T,00], a calibration of the two factor model (8), on the above data yields $\beta(t) = (\frac{d}{dt}E(l_t))/E(l_t)$, with $E(l_t) = p(t)$, $\alpha = 36.7$, $\sigma_1 = 0.336$, $\sigma_2 = 0.317$; $\rho = 0$. We use $S_0 = 12.5$, $K = 13.50$. The "true" value obtained for $n = 800$ is 2.0816 for FGG and DAG and 1.7853 for FTG. The unit for computational cost is in 1000 seconds, the reported errors are to be multiplied by 10^{-4}. The results are reported in Table 4.

Table 4. American Call Option on a Two Factor Model with Multiplicative Noise

Time	FGG			DAG			FTG		
Step	Option	Time	Error	Option	Time	Error	Option	Time	Error
100	2.0869	0.08	53	2.0866	0.09	50	1.7881	0.14	29
200	2.0836	0.63	20	2.0837	0.71	21	1.7863	1.09	11
300	2.0827	2.13	11	0.0827	2.35	10	1.7858	3.54	6
400	2.0823	4.98	6	2.0822	5.53	6	1.7856	8.58	3

5.3 Conclusions

We developed three methods for arranging the tree geometry: (FGG) originated in [HW, 93], as for (DAG) we carried out to the end our interpretation of a foot note suggestion made in [HW, 90]; finally, (FTG) was designed to match the term structures of forward markets and was proposed in [HW, 94 a,b], in the case of linear drifts and without giving any proofs. In this paper, we established the validity of this construction in a more general context. The numerical performance of FGG and DAG are virtually identical. Mixed results are achieved for FTG: for the nonlinear cases (7) and (8) the positions of the median nodes are obtained by the painstaking calculation (13); and in this case, FTG is only slightly faster than the other methods in the one factor case and actually takes longer for the two factor model. Alternatively, in the linear models (3) and (5), the median nodes are revealed by the solution of the ordinary differential equation mentioned after Remark 5. This enhancement allows the FTG to run twice as fast as the other methods in the one factor model and slightly faster in the two factor case. One conclusion is that FTG is extremely effective when the model considered has linear drift and additive noise. Although FTG's performance was slower when the transformed drift is nonlinear, it still is of value. Indeed, we imposed for the reversion level $l(t)$ an exogenous functional form. In practice the expected value of the spot price $\varphi(t)$ is derived from the knowledge of futures prices and market price of risk analysis. In this case, of all the methods considered only the FTG is able to match this expectation.

References

[HW, 90] Hull, John C. and Alan White. 1990. *Valuing Derivative Securities Using the Explicit Finite Difference Method.* J. of Financial and Quantitative Analysis. Vol. 25, No. 1, March. pp. 87-10.

[HW, 93] Hull, John C. and Alan White. 1993. *One-Factor Interest-Rate Models and the Valuation of Interest-Rate Derivative Securities.* J. of Financial and Quantitative Analysis. Vol. 28, No. 2, June. pp. 235-253.

[HW, 94a] Hull, John C. and Alan White. 1994. *Numerical Procedures for Implementing Term Structure Models I: Single-Factor Models.* J. of Derivatives. Fall. pp. 7-16.

[HW, 94b] Hull, John C. and Alan White. 1994. *Numerical Procedures for Implementing Term Structure Models: Two-Factor Models.* J. of Derivatives. Winter. pp. 37-48.

[JW, 00] James, Jessica and Webber, Nick. *Interest Rate Modelling.* Wiley. 2000.

[KP, 99] Kloeden, Peter E. and Eckhard Platen. 1999. *Numerical Solution of Stochastic Differential Equations.* Springer-Verlag.

[LSW, 00] Lari-Lavassani, Ali, Mohamadreza Simchi and Antony Ware. *A Discrete Valuation of Swing Options.* Preprint, 2000.

[P, 97] Pilipovic, Dragana.1997. *Energy Risk: Valuing and Managing Energy Derivatives.* McGraw-Hill.

[T, 2000] Bradley Tifenbach. 2000. *Numerical Methods for Modeling Energy Spot Prices.* MSc Thesis. University of Calgary. 181 pages.

This work is partially funded by grants from the National Science and Engineering Research Council of Canada and MITACS a Canadian Network of Centres of Excellence.

On the Use of Quasi-Monte Carlo Methods in Computational Finance

Christiane Lemieux[1] and Pierre L'Ecuyer[2]

[1] Department of Mathematics and Statistics, University of Calgary,
2500 University Drive N.W., Calgary, AB, T2N 1N4, Canada
lemieux@math.ucalgary.ca
[2] Département IRO, Université de Montréal, C.P. 6128, Succ. Centre-ville,
Montréal, QC, H3C 3J7, Canada
lecuyer@iro.umontreal.ca

Abstract. We give the background and required tools for applying quasi-Monte Carlo methods efficiently to problems in computational finance, and survey recent developments in this field. We describe methods for pricing european path-dependent options, and also discuss problems involving the estimation of gradients and the simulation of stochastic volatility models.

1 Introduction

The Monte Carlo (MC) method has been introduced in finance in 1977, in the pioneering work of Boyle [5]. In 1995, Paskov and Traub published a paper [42] in which they used *quasi-Monte Carlo* (QMC) methods to estimate the price of a collaterized mortgage obligation. The problem they considered was in high dimensions (360) but nevertheless, they obtained more accurate approximations with QMC methods than with the standard MC method. Since then, many people have been looking at QMC methods has a promising alternative for pricing financial products [20,37,53,1,10,3,49]. Researchers studying QMC methods have also been very interested by these advances in computational finance because they provided convincing numerical results suggesting that QMC methods could do better than MC even in high dimensions, a task that was generally believed to be out of reach.

The aim of this paper is to provide the required background and tools for applying QMC methods to computational finance problems. We first review the idea of QMC methods and recall general results about their performance in comparison with the MC method. We give pseudocode for implementing *Korobov rules* [22], which constitute one type of QMC method, and provide references to papers and websites where other constructions (and code) can be found. Different randomizations are also discussed. In Section 3, we describe how randomized QMC methods can be applied for pricing European path-dependent options under the Black-Scholes model. Various methods that can be used in combination with QMC methods to enhance their performance are discussed in Section 4. We conclude in Section 5 by discussing more complex applications such as the simulation of stochastic volatility models.

V.N. Alexandrov et al. (Eds.): ICCS 2001, LNCS 2073, pp. 607–616, 2001.
© Springer-Verlag Berlin Heidelberg 2001

2 Quasi-Monte Carlo Methods

The general problem for which QMC methods have been proposed as an alternative to the MC method is multidimensional numerical integration. Hence for the remainder of this section, we assume the problem under consideration is to evaluate

$$\mu = \int_{[0,1)^t} f(\mathbf{u}) d\mathbf{u},$$

where f is a square-integrable function. Many problems in finance amount to evaluate such integrals, as we discuss in Section 3. To approximate μ, both MC and QMC proceed by choosing a point set $P_n = \{\mathbf{u}_0, \ldots, \mathbf{u}_{n-1}\} \subset [0,1)^t$, and then the average value of f over P_n is computed, i.e., we get

$$Q_n = \frac{1}{n} \sum_{i=0}^{n-1} f(\mathbf{u}_i). \tag{1}$$

In the MC method, the points $\mathbf{u}_0, \ldots, \mathbf{u}_{n-1}$ are independent and uniformly distributed over $[0,1)^t$. In practice, one uses a pseudorandom number generator to choose these points. The idea of QMC methods is to use a more regularly distributed point set, so that a better sampling of the function can be achieved. An important difference with MC is that the set P_n is typically deterministic when a QMC method is applied. Niederreiter presents these methods in detail in his book [35], and describes different ways of measuring the quality of the point sets P_n on which QMC methods rely. More specifically, the goal is to measure how far is the empirical distribution induced by P_n from the uniform distribution over $[0,1)^t$.

Such measures can be useful for providing upper bounds on the deterministic integration error $|Q_n - \mu|$. For example, the *rectangular-star discrepancy* $D^*(P_n)$ looks at the difference (in absolute value) between the volume of a rectangular "box" aligned with the axes of $[0,1)^t$ and having a corner at the origin, and the fraction of points from P_n contained in the box, and then take the maximum difference over all such boxes. Typically, a point set P_n is called a *low-discrepancy point set* if $D^*(P_n) = O(n^{-1} \log^t n)$. For a function of bounded variation in the sense of Hardy and Krause, the integration error $|Q_n - \mu|$ is in $O(n^{-1} \log^t n)$ when P_n is a low-discrepancy point set (see [35,30,31] and the references therein for the details).

This type of upper bound suggests that the advantage of QMC methods over MC, which has a probabilistic error in $O(n^{-1/2})$, will eventually be lost as the dimension t increases, or more precisely, it suggests that QMC methods will require a sample size n too large, for practical purposes, to improve upon MC when t is large. In this context, numerical results showing an improvement of QMC over MC in high dimensions and using a relatively small sample size n [42, 10,32,26] seem hard to explain. To reconciliate this apparent contradiction, two main approaches have been used. First, the study of *randomized* QMC methods [12,38,51,52,41,26] has provided new tools to understand the advantages of QMC over MC. Second, the notion of *effective dimension*, introduced by Paskov [43]

and redefined in [10,18], has been very useful to understand how QMC methods could improve upon MC even in large dimensions, as we now explain.

2.1 Effective Dimension

The effective dimension of a function is linked to its *ANOVA decomposition* [19, 14,41], which rewrites any square-integrable function $f : [0,1)^t \to \mathbb{R}$ as a sum of 2^t components; there is one component f_I per subset I of $\{1,\dots,t\}$, i.e.,

$$f(\mathbf{u}) = \sum_{I \subseteq \{1,\dots,t\}} f_I(\mathbf{u}),$$

and the f_I's are such that $\int_{[0,1)^t} f_I(\mathbf{u}) d\mathbf{u} = 0$ for any nonempty I, and $\int_{[0,1)^t} f_I(\mathbf{u})$
$f_J(\mathbf{u}) d\mathbf{u} = 0$ for any $I \neq J$. Hence this decomposition is orthogonal and we get

$$\sigma^2 \stackrel{\text{def}}{=} \mathrm{Var}(f) = \sum_{I \subseteq \{1,\dots,t\}} \sigma_I^2,$$

where $\sigma_I^2 = \mathrm{Var}(f_I)$. Therefore, the quantity σ_I^2/σ^2 can be used as a measure of the relative importance of the component f_I for explaining the variance of f. If the l-dimensional components with $l \leq s$ contribute to more than $100\alpha\%$ of the variance (i.e., if $\sum_{I:|I|\leq s} \sigma_I^2 \geq \alpha\sigma^2$), then f is said to have an effective dimension of at most s in the *superposition sense* [10,18] in proportion α. Similar definitions can be given for the effective dimension in the *truncation sense* (if $\sum_{I \subseteq \{1,\dots,s\}} \sigma_I^2 \geq \alpha\sigma^2$) [10,18], or in the *successive dimensions sense* (if $\sum_{I \subseteq \{i,i+1,\dots,i+s-1\}} \sigma_I^2 \geq \alpha\sigma^2$) [26].

It is often the case in computational finance that the functions to be integrated have a low effective dimension in some sense. When this happens, it means that even if the function is t-dimensional with t large, a QMC method based on a point set P_n that has good low-dimensional projections (i.e., such that when $|I|$ is small, the projection $P_n(I)$ of P_n over the subspace of $[0,1)^t$ indexed by the coordinates in I is well distributed) can provide an accurate approximation for μ. Hence the success of QMC methods rely on a combination of "tractable" problems (i.e., problems involving functions with a low effective dimension), and point sets having good low-dimensional projections.

Note that in the study of the effective dimension, the variability of f is measured by its variance rather than, e.g., the bounded variation used in the upper bounds discussed earlier. In this context, it seems natural to measure the quality of an estimator for μ by also looking at its variance. This can be achieved for QMC methods if we randomize their associated point set. By doing so, the integration error can be estimated easily. Also, the analysis of the variance has the advantage of requiring much weaker conditions on the function f than when the deterministic error is studied.

2.2 Constructions

Two main families of QMC methods are the *lattice rules* and the *digital nets* [35, 47]. Korobov rules are a special case of lattice rules that are easy to implement, as we now describe. For a given sample size n, the only parameter required to generate a point set P_n in t dimensions is an integer a relatively prime to n. We then get

$$P_n = \left\{ \frac{i}{n}(1, a, a^2, \ldots, a^{t-1}) \bmod 1,\ i = 0, \ldots, n-1 \right\}, \tag{2}$$

where the modulo 1 is applied component-wise. The choice of the generator a is important and tables of values of a leading to point sets P_n that are "good" for many values of t are given in [26]. What do we mean by good ? The criterion used to measure the quality of P_n in [26] looks at many low-dimensional projections of the point set P_n and makes sure they are well distributed (with respect to the *spectral test*), in agreement with the requirements mentioned in the previous subsection. The points in P_n can be generated very easily as follows:

`input:` a, n, t
$g_0 = 1$
`for` $j = 1$ `to` $t-1$ `do` $g_j = (g_{j-1} \times a) \bmod n$
$\mathbf{u} = (0, \ldots, 0)$
`for` $i = 1$ `to` $n-1$ `do` $\mathbf{u} = (\mathbf{u} + (g_0/n, \ldots, g_{t-1}/n)) \bmod 1$

The generation of the points in (2) can be done in an even simpler and faster way than that illustrated above when n is prime and a is a primitive element modulo n; see [26] for more details. In any case, generating the point set P_n is faster than when MC is used, and this holds for most QMC methods.

Two nice properties of the point set (2) are that it is *dimension-stationary* and *fully projection-regular* [26,47]. The first property means that if two subsets $I = \{i_1, \ldots, i_s\}, J = \{j_1, \ldots, j_s\}$ of equal cardinality are such that $j_l - i_l$ is constant for $l = 1, \ldots, s$, then $P_n(I) = P_n(J)$, i.e., the projection of P_n over the subspaces of $[0,1)^t$ indexed by the the coordinates in I and J is the same. For example, it means that all the two-dimensional projections of the form $P_n(\{j, j+1\})$, for $j = 1, \ldots, t-1$, are the same. Not all QMC methods have this property. The second property simply means that all projections of P_n have n distinct points, which is certainly desirable.

Another construction that shares many similarities with lattice rules are the *polynomial lattice rules* [25,29]. As explained in [33,34,26], special cases of both methods can be constructed by using all overlapping t-tuples output by a *linear congruential generator* and a *Tausworthe generator*, respectively, from all possible initial seeds.

As for digital nets, details on their construction can be found in [35] and the references therein. Improved constructions are presented in, e.g., [50,45,36, 46,11,44]. Details on the implementation of Sobol's sequence [48], and Faure's

sequence [15], which were the first constructions proposed in the family of digital nets, are given in [7] and [16], respectively. The code that goes with these two papers can be found at `www.acm.org/calgo/`. More recent software for these methods and other ones can be found at `www.mathdirect.com/products/qrn/` and `www.cs.columbia.edu/~ap/html/finder.html`, which is the link to the FinDer software [42]. The MC and QMC methods' website `www.mcqmc.org` contains other relevant links.

2.3 Randomizations

As mentioned earlier, it is often useful for the purpose of error estimation to randomize QMC point sets. Two desirable properties that a given randomization should have are: (1) each point in the randomized point set should have a uniform distribution on $[0,1)^t$; (2) the regularity of the point set should be preserved. The three randomizations discussed below have these properties.

For lattice rules, Cranley and Patterson [12] suggested to randomly generate a vector Δ in $[0,1)^t$, and then add it to each point of P_n, modulo 1. This means that in the pseudocode given above, before the loop over i, one simply needs to call a pseudorandom generator t times to generate the vector $\Delta = (\Delta_1, \ldots, \Delta_t)$, and then output $(\mathbf{u} + \Delta) \bmod 1$ instead of $\mathbf{u}$. The variance of the estimator

$$\frac{1}{n}\sum_{i=0}^{n-1} f((\mathbf{u}_i + \Delta) \bmod 1)$$

based on a randomly shifted lattice rule is studied in [26,52]; in [26], the only condition required on f is that it must be square-integrable. This randomization can be applied to other types of QMC point sets, as suggested in [51]. However, for digital nets and polynomial lattice rules, using a "XOR-shift" as proposed by Raymond Couture [25,29] is more natural because it preserves the equidistribution properties of this type of point sets. The idea is to generate a random vector Δ in $[0,1)^t$, but instead of adding it to each point $\mathbf{u}_i = (u_{i1}, \ldots, u_{it})$ of P_n modulo 1, an exclusive-or operation between the binary representation of Δ_j and u_{ij} is performed, for each dimension $j = 1, \ldots, t$. The variance of the estimator based on a polynomial lattice rule that has been XOR-shifted is studied in [29]. Another randomization that can be used for those point sets is the *scrambling* of Owen [38], which leads to tighter bounds on the variance of the associated estimators [38,39,40], but it requires more computation time than the XOR-shift.

3 Pricing under the Black-Scholes Model

In this section, we describe how to use QMC methods for estimating the value of a financial contract, such as an option, whose underlying assets follow the Black-Scholes model [4]. More precisely, we assume the goal is to estimate

$$\mu = \mathrm{E}^*(g_{\mathbf{p}}(S_1, \ldots, S_t)),$$

where $S_1, \ldots, S_t$ are prices (e.g., from one asset at t different times, or from t assets at the expiration date of the contract) that have a lognormal distribution. The function $g_{\mathbf{p}}$ is assumed to be square-integrable and it represents the discounted payoff of the contract, and $\mathbf{p}$ is a vector of parameters (e.g., containing the risk-free rate r, the volatility σ, the strike price K, the expiration time T, etc.). The expectation is taken under the risk-neutral measure [13]. Written similarly as in (1), the MC estimator for μ is given by

$$\frac{1}{n}\sum_{i=0}^{n-1} f_{\mathbf{p}}(\mathbf{u}_i), \tag{3}$$

where the $\mathbf{u}_i$'s are independent and uniformly distributed over $[0,1)^t$, and $f_{\mathbf{p}} : [0,1)^t \to \mathbb{R}$ is a function that takes as an input a sequence of t numbers $u_1, \ldots, u_t$ between 0 and 1, transforms them into observations of $S_1, \ldots, S_t$, and then evaluates $g_{\mathbf{p}}(S_1, \ldots, S_t)$. Also, $f_{\mathbf{p}}$ is such that $\mathrm{E}^*(f_{\mathbf{p}}(\mathbf{u}_i)) = \int_{[0,1)^t} f_{\mathbf{p}}(\mathbf{u}) d\mathbf{u} = \mu$. For example, if f represents the discounted payoff of a path-dependent option on one asset, then $S_1, \ldots, S_t$ would represent observed prices on one path of this asset. To generate these prices, start with u_1, transform it into an observation x from the standard normal distribution (using inversion, see, e.g., [23]), and then generate the first price by letting $S_1 = S_0 e^{rt_1 + \sigma\sqrt{t_1}x}$, where t_1 is the time at which S_1 is observed, and S_0 is the price of the underlying asset at time 0. In a similar way, u_2 can be used to generate the second price S_2, and so on. The precise definition of $f_{\mathbf{p}}$ for an Asian option pricing problem is given in [26]. In the case where one has to generate prices of correlated assets, procedures requiring one uniform number per observed price can be found in, e.g., [1].

The QMC estimator for μ can be built in the exact same way as for MC if we use a randomized QMC point set: just take the estimator (3) but with the $\mathbf{u}_i$'s coming from a randomized QMC point set. With an appropriate randomization, each point $\mathbf{u}_i$ has a uniform distribution over $[0,1)^t$ and thus the observation $f_{\mathbf{p}}(\mathbf{u}_i)$ has the same distribution as in the MC setting. Hence (3) is an unbiased estimator of μ in both cases. The only difference with MC is that with QMC, the observations $f_{\mathbf{p}}(\mathbf{u}_0), \ldots, f_{\mathbf{p}}(\mathbf{u}_{n-1})$ are correlated instead of being independent. With a carefully chosen QMC point set, the induced correlation should be such that the estimator has a smaller variance than the MC estimator. The variance of the randomized QMC estimator can be estimated by constructing M i.i.d. copies of the estimator (3) (e.g., with M i.i.d. random shifts), and then computing the sample variance.

4 Reducing the Variance and/or the Dimension

To increase the efficiency of MC simulations, many variance reduction techniques are available [24,23] and can be applied for financial simulations; see the survey [6] for an overview. A good example of such technique is for pricing an Asian option on the arithmetic average; one can then use as a *control variable* the price of the same option but taken on the *geometric* average [21]. This can reduce the

variance by very large factors. These techniques can usually be combined with (randomized) QMC methods in a straightforward way; see, e.g., [10,53,26]. The combination almost always improves the naive QMC estimators, but usually the advantage of QMC over MC is decreased by applying variance reduction techniques. Intuitively, this can be explained by the fact that these techniques might reduce the variance of the function in a way that concentrates the remaining variance in very small regions, and this makes it harder for QMC to improve upon MC.

Techniques for reducing the effective dimension are useless to improve MC simulations, but they can greatly enhance the performance of QMC methods [9, 32,1,10,27,2]. We discuss two of them: the *Brownian bridge* (BB) and the *principal components* (PC) techniques. The idea of BB was first introduced in [9] and can be used to generate a Brownian motion at T different times $B(t_1), \ldots, B(t_T)$ by using T uniform numbers $u_1, \ldots, u_T$. Instead of generating these observations sequentially (as outlined in the previous section), u_1 is used to generate $B(t_T)$, u_2 is used to generate $B(t_{\lfloor T/2 \rfloor})$, u_3 for $B(t_{\lfloor T/4 \rfloor})$, u_4 for $B(t_{\lfloor 3T/4 \rfloor})$, etc. This can be done easily since for $u < v < w$, the distribution of $B(v)$ given $B(u)$ and $B(w)$ is Gaussian with parameters depending only on u, v, w. The reason why this can be helpful for QMC methods is that by generating the Brownian motion path in this way, more importance is given to the first few uniform numbers, and thus the effective dimension of a function depending on $B(t_1), \ldots, B(t_T)$ should be decreased by doing that.

In the same spirit as BB, one can decompose the variance-covariance matrix of the prices to be simulated using principal components, and then generate the prices using this decomposition [1]. An advantage of this approach over BB is that it can be used to generate prices of correlated assets whereas BB can only be applied for generating prices coming from a single path. However, PC requires more computation time than BB for the generation of the prices, but see [2] for a way of speeding up PC.

5 Broadening the Range of Applications

We conclude this paper by discussing applications that go beyond the context discussed in Section 3. One such application is the estimation of derivatives (or gradients) of prices with respect to one (or more) parameter(s) (which are often called *the greeks*). For example, one might be interested in estimating how sensitive an option's price is to the volatility of the underlying asset. Broadie and Glasserman [8] discuss how to do this using MC combined with some variance reduction techniques; a QMC approach based on randomly shifted lattice rules that improves upon MC is presented in [28]. These estimators could also be used for the more complex problem of American option pricing since the latter can be addressed using *stochastic approximation* methods that require gradient estimators, as discussed in [17].

QMC estimators can also be used for pricing contracts that depend on assets whose volatility is assumed to be stochastic. The difference with the problems

discussed in Section 3 is that one needs to discretize the price process and at each time step in the discretization, an observation from the volatility process must be generated in addition to one from the asset's price. Hence for T time steps, at least $2T$ random numbers are required to generate one path, which means the dimension of the problem is also at least $2T$. When using QMC, such simulations require a careful assignment of the uniform numbers $u_1, \ldots, u_{2T}$ to the generation of the prices and the volatilities [3], but improvement upon MC can still be achieved in this context [53,3].

These applications are just a small sample of the possible problems for which QMC can provide more precise estimators than MC in computational finance. We believe that basically any problem that can be addressed using MC also has a QMC counterpart that can not only reduce the variance of the estimators, but that also typically requires less computation time.

References

1. P. Acworth, M. Broadie, and P. Glasserman. A comparison of some Monte Carlo and quasi-Monte Carlo techniques for option pricing. In P. Hellekalek and H. Niederreiter, editors, *Monte Carlo and Quasi-Monte Carlo Methods in Scientific Computing*, number 127 in Lecture Notes in Statistics, pages 1–18. Springer-Verlag, 1997.
2. F. Åkesson and J. P. Lehoczy. Path generation for quasi-Monte Carlo simulation of mortgage-backed securities. *Management Science*, 46:1171–1187, 2000.
3. H. Ben Ameur, P. L'Ecuyer, and C. Lemieux. Variance reduction of Monte Carlo and randomized quasi-Monte Carlo estimators for stochastic volatility models in finance. In *Proceedings of the 1999 Winter Simulation Conference*, pages 632–639. IEEE Press, December 1999.
4. F. Black and M. Scholes. The pricing of options and corporate liabilities. *Journal of Political Economy*, 81:637–654, 1973.
5. P. Boyle. Options: a Monte Carlo approach. *Journal of Financial Economics*, 4:323–338, 1977.
6. P. Boyle, M. Broadie, and P. Glasserman. Monte Carlo methods for security pricing. *Journal of Economic Dynamics & Control*, 21(8-9):1267–1321, 1997.
7. P. Bratley and B. L. Fox. Algorithm 659: Implementing Sobol's quasirandom sequence generator. *ACM Transactions on Mathematical Software*, 14(1):88–100, 1988.
8. M. Broadie and P. Glasserman. Estimating security price derivatives using simulation. *Management Science*, 42:269–285, 1996.
9. R. E. Caflisch and B. Moskowitz. Modified Monte Carlo methods using quasi-random sequences. In H. Niederreiter and P. J.-S. Shiue, editors, *Monte Carlo and Quasi-Monte Carlo Methods in Scientific Computing*, number 106 in Lecture Notes in Statistics, pages 1–16, New York, 1995. Springer-Verlag.
10. R. E. Caflish, W. Morokoff, and A. B. Owen. Valuation of mortgage-backed securities using Brownian bridges to reduce effective dimension. *The Journal of Computational Finance*, 1(1):27–46, 1997.
11. A.T. Clayman, K.M. Lawrence, G.L. Mullen, H. Niederreiter, and N.J.A. Sloane. Updated tables of parameters of (t, m, s)-nets. *Journal of Comb. Designs*, 7:381–393, 1999.

12. R. Cranley and T. N. L. Patterson. Randomization of number theoretic methods for multiple integration. *SIAM Journal on Numerical Analysis*, 13(6):904–914, 1976.
13. D. Duffie. *Dynamic Asset Pricing Theory*. Princeton University Press, second edition, 1996.
14. B. Efron and C. Stein. The jackknife estimator of variance. *Annals of Statistics*, 9:586–596, 1981.
15. H. Faure. Discrépance des suites associées à un système de numération. *Acta Arithmetica*, 61:337–351, 1982.
16. B. L. Fox. Implementation and relative efficiency of quasirandom sequence generators. *ACM Transactions on Mathematical Software*, 12:362–376, 1986.
17. M.C. Fu, S.B. Laprise, D.B. Madan, Y. Su, and R. Wu. Pricing american options: A comparison of Monte Carlo simulation approaches. *Journal of Computational Finance*, 2:49–74, 1999.
18. F. J. Hickernell. Lattice rules: How well do they measure up? In P. Hellekalek and G. Larcher, editors, *Random and Quasi-Random Point Sets*, volume 138 of *Lecture Notes in Statistics*, pages 109–166. Springer, New York, 1998.
19. W. Hoeffding. A class of statistics with asymptotically normal distributions. *Annals of Mathematical Statistics*, 19:293–325, 1948.
20. C. Joy, P. P. Boyle, and K. S. Tan. Quasi-Monte Carlo methods in numerical finance. *Management Science*, 42:926–938, 1996.
21. A. G. Z. Kemna and A. C. F. Vorst. A pricing method for options based on average asset values. *Journal of Banking and Finance*, 14:113–129, 1990.
22. N. M. Korobov. The approximate computation of multiple integrals. *Dokl. Akad. Nauk SSSR*, 124:1207–1210, 1959. in Russian.
23. A. M. Law and W. D. Kelton. *Simulation Modeling and Analysis*. McGraw-Hill, New York, third edition, 2000.
24. P. L'Ecuyer. Efficiency improvement via variance reduction. In *Proceedings of the 1994 Winter Simulation Conference*, pages 122–132. IEEE Press, 1994.
25. P. L'Ecuyer and C. Lemieux. Quasi-Monte Carlo via linear shift-register sequences. In *Proceedings of the 1999 Winter Simulation Conference*, pages 336–343. IEEE Press, 1999.
26. P. L'Ecuyer and C. Lemieux. Variance reduction via lattice rules. *Management Science*, 46:1214–1235, 2000.
27. C. Lemieux and P. L'Ecuyer. A comparison of Monte Carlo, lattice rules and other low-discrepancy point sets. In H. Niederreiter and J. Spanier, editors, *Monte Carlo and Quasi-Monte Carlo Methods 1998*, pages 326–340, Berlin, 2000. Springer.
28. C. Lemieux and P. L'Ecuyer. Using lattice rules for variance reduction in simulation. In *Proceedings of the 2000 Winter Simulation Conference*, pages 509–516, Piscataway, NJ, 2000. IEEE Press.
29. C. Lemieux and P. L'Ecuyer. Polynomial lattice rules. In preparation, 2001.
30. W. J. Morokoff and R. E. Caflisch. Quasi-random sequences and their discrepancies. *SIAM Journal on Scientific Computing*, 15:1251–1279, 1994.
31. W. J. Morokoff and R. E. Caflish. Quasi-Monte Carlo integration. *Journal of Computational Physics*, 122:218–230, 1995.
32. W. J. Morokoff and R. E. Caflish. Quasi-Monte Carlo simulation of random walks in finance. In P. Hellekalek and H. Niederreiter, editors, *Monte Carlo and Quasi-Monte Carlo Methods in Scientific Computing*, number 127 in Lecture Notes in Statistics, pages 340–352. Springer-Verlag, 1997.
33. H. Niederreiter. Quasi-Monte Carlo methods and pseudorandom numbers. *Bulletin of the American Mathematical Society*, 84(6):957–1041, 1978.

34. H. Niederreiter. Multidimensional numerical integration using pseudorandom numbers. *Mathematical Programming Study*, 27:17–38, 1986.
35. H. Niederreiter. *Random Number Generation and Quasi-Monte Carlo Methods*, volume 63 of *SIAM CBMS-NSF Regional Conference Series in Applied Mathematics*. SIAM, Philadelphia, 1992.
36. H. Niederreiter and C. Xing. Nets, (t, s)-sequences, and algebraic geometry. In P. Hellekalek and G. Larcher, editors, *Random and Quasi-Random Point Sets*, volume 138 of *Lecture Notes in Statistics*, pages 267–302. Springer, New York, 1998.
37. S. Ninomiya and S. Tezuka. Toward real-time pricing of complex financial derivatives. *Applied Mathematical Finance*, 3:1–20, 1996.
38. A. B. Owen. Randomly permuted (t, m, s)-nets and (t, s)-sequences. In H. Niederreiter and P. J.-S. Shiue, editors, *Monte Carlo and Quasi-Monte Carlo Methods in Scientific Computing*, number 106 in Lecture Notes in Statistics, pages 299–317. Springer-Verlag, 1995.
39. A. B. Owen. Monte Carlo variance of scrambled equidistribution quadrature. *SIAM Journal on Numerical Analysis*, 34(5):1884–1910, 1997.
40. A. B. Owen. Scrambled net variance for integrals of smooth functions. *Annals of Statistics*, 25(4):1541–1562, 1997.
41. A. B. Owen. Latin supercube sampling for very high-dimensional simulations. *ACM Transactions of Modeling and Computer Simulation*, 8(1):71–102, 1998.
42. S. Paskov and J. Traub. Faster valuation of financial derivatives. *Journal of Portfolio Management*, 22:113–120, 1995.
43. S.P. Paskov. New methodologies for valuing derivatives. In S. Pliska and M. Dempster, editors, *Mathematics of Derivative Securities*. Cambridge University Press, Isaac Newton Institute, Cambridge, 1996.
44. G. Pirsic and W. Ch. Schmid. Calculation of the quality parameter of digital nets and application to their construction. *J. Complexity*, 2001. To appear.
45. W. Ch. Schmid. Shift-nets: a new class of binary digital (t, m, s)-nets. In P. Hellekalek, G. Larcher, H. Niederreiter, and P. Zinterhof, editors, *Monte Carlo and Quasi-Monte Carlo Methods in Scientific Computing*, volume 127 of *Lecture Notes in Statistics*, pages 369–381, New York, 1997. Springer-Verlag.
46. W. Ch. Schmid. Improvements and extensions of the "Salzburg Tables" by using irreducible polynomials. In H. Niederreiter and J. Spanier, editors, *Monte Carlo and Quasi-Monte Carlo Methods 1998*, pages 436–447, Berlin, 2000. Springer.
47. I. H. Sloan and S. Joe. *Lattice Methods for Multiple Integration*. Clarendon Press, Oxford, 1994.
48. I. M. Sobol'. The distribution of points in a cube and the approximate evaluation of integrals. *U.S.S.R. Comput. Math. and Math. Phys.*, 7:86–112, 1967.
49. K. S. Tan and P.P. Boyle. Applications of randomized low discrepancy sequences to the valuation of complex securities. *Journal of Economic Dynamics and Control*, 24:1747–1782, 2000.
50. S. Tezuka. *Uniform Random Numbers: Theory and Practice*. Kluwer Academic Publishers, Norwell, Mass., 1995.
51. B. Tuffin. On the use of low-discrepancy sequences in Monte Carlo methods. Technical Report No. 1060, I.R.I.S.A., Rennes, France, 1996.
52. B. Tuffin. Variance reduction order using good lattice points in Monte Carlo methods. *Computing*, 61:371–378, 1998.
53. G. A. Willard. Calculating prices and sensitivities for path-dependent derivatives securities in multifactor models. *Journal of Derivatives*, 5:45–61, Fall 1997.

An Efficient Algorithm to Calculate the Minkowski Sum of Convex 3D Polyhedra

Henk Bekker and Jos B.T.M. Roerdink

Institute for Mathematics and Computing Science, University of Groningen,
P.O.B. 800 9700 AV Groningen, The Netherlands,
{bekker,roe}@cs.rug.nl

Abstract. A new method is presented to calculate the Minkowski sum of two convex polyhedra A and B in 3D. The method works as follows. The slope diagrams of A and B are considered as graphs. These graphs are given edge attributes. From these attributed graphs the attributed graph of the Minkowski sum is constructed. This graph is then transformed into the Minkowski sum of A and B. The running time of the algorithm is linear in the number of edges of the Minkowski sum.

1 Introduction: The Minkowski Sum and the Slope Diagram

The Minkowski sum of two sets $A, B \subseteq R^3$ is defined as

$$A \oplus B = \{a + b | a \in A, b \in B\}. \tag{1}$$

In this article A and B are convex polyhedra in R^3, and we represent their Minkowski sum by C, so, $C = A \oplus B$. It can be easily shown that C is also a convex polyhedron, but in general C is more complex then A and B. E.g. the faces of C consist of all the faces of A and B, and some additional faces. See fig. 1a, 1b, 1f, 3b. The Minkowski sum can be defined in a space of any dimension. Amongst others, it is used in computational geometry, computer vision and imaging, robot motion planning and in pattern recognition. Our motivation for designing an efficient Minkowski sum algorithm comes from mathematical morphology. In this field we are experimenting with a method to compare the shape of two convex polyhedra, based on Minkowski addition [1, 2]. In this method, to calculate the similarity of two convex polyhedra, their Minkowski sum has to be calculated for many relative orientations (hundreds) of the polyhedra.

In 2D space, algorithms are known [4] to compute the Minkowski sum of two convex polygons A and B in linear time $O(nv_A + nv_B)$, where nv_A, nv_B are the number of vertices of A, B respectively. In R^3 two classes of algorithms exist to compute the Minkowski sum of two convex polyhedra; the ones working in R^3, and the ones working in *slope diagram space*. We will denote these two classes by MSR and MSD. In essence, MSD methods work in two dimensional space.

V.N. Alexandrov et al. (Eds.): ICCS 2001, LNCS 2073, pp. 619–628, 2001.
© Springer-Verlag Berlin Heidelberg 2001

As we will see later, in MSD methods the polyhedra A and B are transformed to a 2D space. There the transformed polyhedra A and B are added in some way, and the result is backtransformed, giving C. In general, MSR algorithms are simpler to implement than MSD algorithms, but are less efficient.

In the literature much is said about MSR algorithms but hardly any integral and concrete discussion of MSD algorithms is available. In [3] it is shown that it is in principle possible to calculate C in linear time (Later we explain the meaning of linear in more detail). However, no concrete method or algorithm is given. In this article we discuss briefly three known algorithms, called method1..method3, and present our own algorithm, method4. Method1 is a simple and expensive MSR method. Method2 is a mixed MSR-MSD method. It is complex and not efficient. Method3 is a generic MSD method but we think it has a time complexity that is worse than the one derived in [3]. Method4 is an MSD method with a linear time complexity, and is easy to implement. Before discussing these methods we introduce our representation of polyhedra, and introduce the slope diagram.

We represent a convex polyhedron, say A, by an attributed graph. Nodes, edges and faces of this graph represent vertices, edges and faces resp. of A. Every node of the graph has an attribute representing the position of the corresponding vertex. In this paper, a polyhedron and its graph are equivalent, so, calculating the Minkowski sum of two convex polyhedra A and B is equivalent to transforming the attributed graphs A and B into an attributed graph C.

The graphs representing polyhedra are so-called *polygonal* graphs. They have the property that they are plane, and that every edge bounds two different faces. (The outer region of the graph is also a face.) A polygonal graph A may be transformed into another polygonal graph, its *dual* graph, denoted by $dual(A)$ or DA. DA is calculated as follows:

- DA has one node for each face f of A, denoted by $dual(f)$.
- DA has one edge for each edge of A. Let e be a common edge of the faces f_i and f_j of A. Then in DA the nodes $dual(f_i)$ and $dual(f_j)$ are connected by an edge, called the dual of e.

It can be easily checked that in this way the nodes of A give faces of DA. Clearly, by computing $dual(A)$ only the graph structure of DA is defined, not its attributes. For a polygonal graph A it holds that $dual(DA) = A$

A drawing of a graph on some surface (e.g. the plane or a sphere) such that no two edges cross is called an embedding of the graph. We now introduce the embedding on the unit sphere of the graphs DA and DB. We call these embeddings SDA and SDB, or the *slope diagrams* of A and B. To compute SDA, (and similarly SDB) we have to define where every node and edge of DA is mapped on the sphere. First consider the nodes. Every node n of DA is the image of some face f of A. To n is assigned as attribute the outward unit normal on the face f. The node n is mapped on the sphere to the end point of this unit vector. Second consider the edges. An edge e connecting in DA the nodes n_1 and n_2 is mapped to the arc of the unit circle on the sphere connecting the images

of n_1 and n_2. For an example of two polyhedra and their slope diagrams, see fig. 1a, 1b, 1c, 1d.

A few words about designating the elements of the slope diagram. A slope diagram consists of spherical faces, spherical edges and points on a sphere. In the rest of this article we omit the word "spherical", so we will speak about the faces, edges and points of a slope diagram. So, a face, edge or point of a slope diagram is the image of a vertex, edge or face resp. of the corresponding polyhedron.

From SDA and SDB a new slope diagram may be created by *overlaying* SDA and SDB. Overlaying two embedded graphs amounts roughly speaking to superimposing the two graphs and merging them into one graph [6, 7]. An important well known property of the Minkowski sum is that the slope diagram of C is identical to the overlay of the slope diagrams of A and B [2], so,

$$SDC = overlay(SDA, SDB).$$

The node positions of SDC consist of (i) the node positions of SDA and SDB, and (ii) the node positions defined by intersecting edges of SDA and SDB. The first ones may be copied from SDA and SDB to SDC, the latter ones are obtained during the overlay calculation. It is important to note that by calculating the dual of SDC the graph structure of C is obtained, but because this graph has no node attributes, no complete description of C is available yet. In a later section we show how in method3 and method4 the attributes of SDC are calculated.

2 Some Common Methods to Calculate the Minkowski Sum

Method1 is a pure MSR method; it is simple but time consuming [5]. It is a two step process. In the first step the position vectors of all the vertices of A are added to the position vectors of all the vertices of B. This results in a total of $nv_A nv_B$ points where nv_A and nv_B are the number of vertices of A and B resp. In the second step the convex hull of these points is computed, giving C. Obviously, the first step has time complexity $O(nv_A nv_B)$. Using some standard convex hull algorithm [4] the second step has time complexity $O(nv_A nv_B \log(nv_A nv_B))$. This method of computing C is expensive because it works entirely in R^3, whereas using SDA and SDB implies working in R^2. Another disadvantage is that the result is not a graph but a set of points. Yet, this method is often used when efficiency is not crucial.

Method2 is a mixed MSR-MSD method. The key idea of this method is to compute all planes bounding C, i.e. all planes that contain a face of C. By calculating the intersections of these planes, the edges and vertices of C are computed. The method works as follows.

1. For every face f of A it is determined in which face of SDB the slope diagram image of f is located. This face of SDB is the image of some vertex of B,

say v. Now the plane containing the face f is translated over the position vector of v. The resulting plane is a bounding plane of C.
2. The same as 1 with A and B interchanged.
3. In the superimposed slope diagrams of A and B it is determined which edges of SDA intersect edges of SDB. Assume that the edges se_i and se_j intersect. Assume that the corresponding edges in A and B are e_i and e_j. Now we construct a plane containing e_i that is parallel with e_j. This plane is shifted over a vector ending somewhere on e_j (say one of its endpoints). The resulting plane is a bounding plane of C.

The intersection of the half spaces defined by the planes described above is C. The faces of C contained in the planes as constructed in step 1 and 2, have the same shape and size as the faces of A resp. B, i.e. are shifted instances of the faces of A resp. B. The faces of C contained in the planes constructed in 3 are new faces, i.e. are not copies of the faces of A or B. These faces are parallelograms with edges e_i and e_j. See figure 1f, 3b for examples. Method2 is more efficient than method1 because it uses slope diagrams. Yet, it contains much redundant work: C contains many faces identical with faces of A and B, but this fact is not used in this method. Most faces are completely reconstructed.

Concluding: in both methods too much geometrical computations are done. The method we propose aims at minimizing these geometrical computations.

3 The Minkowski Sum by Merging Attributed Graphs

Method3 is a straightforward MSD method, but in the literature we could not find an integral description of it. It consists of the following four steps.

1. Calculate the slope diagram SDA. Besides the earlier mentioned node attributes (unit vectors), the slope diagram is given face attributes. Every face f of SDA is given an attribute $attr(f)$, namely the position vector of the corresponding vertex in A. The attributed slope diagram SDB is calculated similarly.
2. Calculate the overlay of SDA and SDB, that is, calculate the graph of SDC. This graph has no attributes yet.
3. Calculate the face attributes of SDC. This is done as follows. When SDA, SDB and SDC are superimposed, every face f of SDC is located in precisely one face f_i of SDA, and in precisely one face f_j of SDB. Face f gets as attribute the sum of the attributes of f_i and f_j.
4. Calculate the dual graph C of SDC as follows. Copy from SDC the face attributes to the corresponding nodes of C. The graph C, with its node attributes, represents the Minkowski sum of A and B.

In the following, the process of determining for every face of SDC in which face of SDA and SDB it is located (see point 3), will be called *face location*. It is instructive to compare method3 with method1. In method1 all vertices of A are combined with all vertices of B. Afterwards, during the convex hull computation,

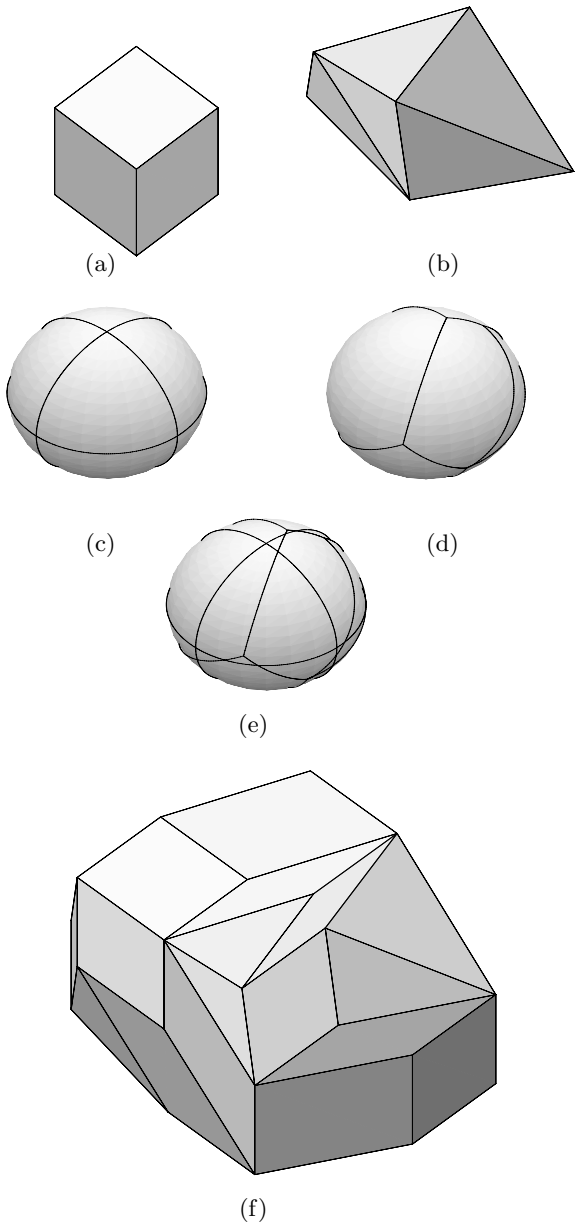


Fig. 1. Two polyhedra A and B (a) (b), their slope diagrams SDA and SDB (c), (d), the overlay of these slope diagrams SDC (e), and the Minkowski sum C of the polyhedra A and B, (f). It may take some time to see the relation between (b) and (d). It can be seen that the (f) consists of the faces of (a) and (b), and additional parallelogram faces (See also figure 3b). MSR methods calculate (f) directly from (a) and (b). MSD methods use the slope diagrams in (c), (d) and (e) to calculate (f).

it is decided which of these points are vertices of C. In method3, by face location, it is decided which vertices of A and B have to be combined to give a vertex of C.

Using a standard graph method [8], the time complexity of calculating SDA and SDB is $O(ne_A+ne_B)$ where ne_A and ne_B are the number of edges of A and B. In the next section we show that calculating the overlay of SDA and SDB can be done in time $O(ne_A + ne_B + k)$ where k is the number of intersecting edges of SDA and SDB. Method3 may be summarized as follows. Step 1 and 4 are transformations to and from the slope diagram domain. In step 2 the overlay is constructed, and in step 3 the face attributes of SDC are calculated. In the following sections we will take a closer look at calculating the overlay and face location.

4 Overlaying and Face Location

Overlaying two subdivisions of the plane is a standard problem of computational geometry. Unfortunately, for our problem, i.e. calculating the overlay of two subdivision of the sphere, no implementations are available, so we had to develop our own implementation. For this we adapted an existing implementation in the plane [6, 7] that runs in linear time $O(ne_A + ne_B + k)$, where k is the number of intersecting edges of SDA and SDB. An additional feature of our implementation is that the edges in the overlay SDC get an attribute indicating from which edge of SDA or SDB the edge stems. Let us explain. When SDA, SDB and SDC are superimposed, every edge e of SDC coincides with part of or a whole edge of SDA or SDB (that is roughly speaking the definition of an overlay). During face location, it is necessary to know which edges of SDC bound a given face of SDA or SDB. Therefore, during the overlay construction, every edge of SDC is given two attributes, one referring to an edge of SDA and one referring to an edge of SDB. In general, only one of these references is non-nil. Only when an edge of SDA (partially) coincides with an edge of SDB, both references are non-nil. This situation occurs for example in the extreme case when the Minkowski sum of two identical polyhedra is calculated, i.e. when $C = A \oplus A$.

Now consider face location. The edge attributes of SDC as described above make it possible to find in SDC the edges that coincide with the edges of SDA or SDB, and that bound a given face of SDA or SDB. We call such a set of edges an A-cycle or a B-cycle. Let us look at some A-cycle a. See fig. 2(c). We want to find all faces of SDC that are within a. First we collect those nodes of SDC that are on or within a. Let us call the set of nodes on and inside a, $a.nodes$. Nodes on a are found by going through the edges of a. Nodes within a are found when, starting from every node on a, inward edges are followed recursively. Using the $a.nodes$, we collect all faces of SDC that have one or more nodes of $a.nodes$ as vertex. The faces collected in this way are within or directly adjacent to a. From these faces those ones are selected that only have vertices from $a.nodes$. These faces are inside a, and get an attribute referring to

the face of SDA corresponding to a. This is done for all A-cycles and B-cycles. In this way every face of SDC gets two attributes telling in which face of SDA and SDB it is located. Using these attributes, every face of SDC is given a vector valued attribute in the following way. If face f of SDC is in face f_i of SDA and in face F_j of SDB then face f gets a vector attribute $attr(f) = attr(f_i) + attr(f_j)$. In LEDA [8], the Computational Geometry platform we use, all operations in the face location algorithm above are available as standard methods.

We will not discuss the time complexity of face location. In the first place because it is not trivial, and second because in the in the next section we present our MSD method, i.e. method4, that works without face location. Because the time complexity of method4 is dominated by calculating the overlay, method4 is superior to method3, whatever the time complexity of face location in method3 may be.

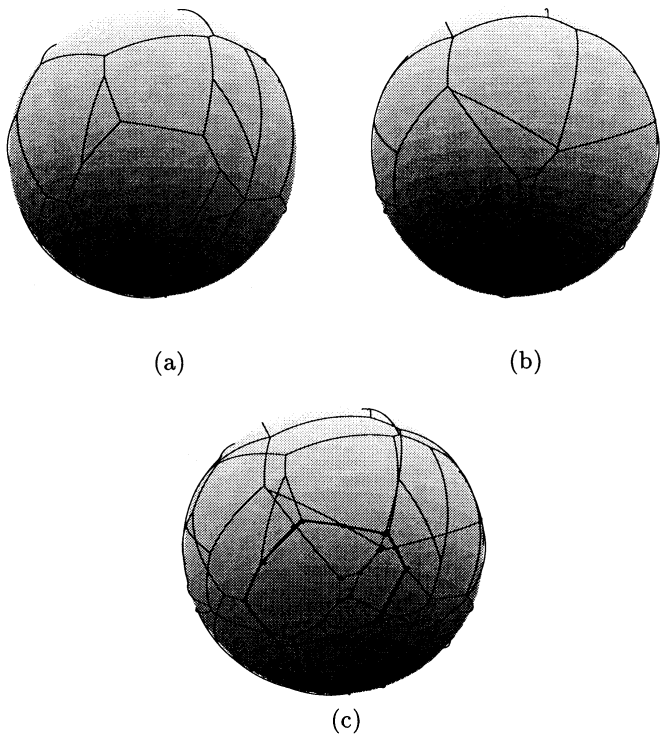


Fig. 2. The slope diagrams SDA (a) and SDB (b) of two randomly generated polyhedra, and the overlay SDC (c) of these slope diagrams. In SDC an A-cycle a is shown. The nodes on and inside a are marked.

5 A Method without Face Location

In method 3 face location was essential for calculating the face attributes of SDC, i.e. for calculating the vertex positions of C. We now present method4, which works with edge attributes instead of face attributes, and thus avoids face location. As a side effect, the absolute position of C is lost. However, in a final step this position is recovered. To describe and implement method4 we use bidirected graphs. This means that every edge e of A, B, C, SDA, SDB and SDC has a *source node* and a *target node*, designated by $source(e)$ resp. $target(e)$. Moreover, for every edge there is a reversal edge, i.e. when there exists an edge e starting at $source(e)$ and ending at $target(e)$ then there is also an edge starting at $target(e)$ and ending at $source(e)$.

Method4 is a six step process and works as follows.

1. Switch to relative coordinates of A and B. More precisely, instead of using node attributes representing absolute node positions, we switch to edge attributes. Each edge of A and B is attributed with a 3D vector. The vector is the relative position of the target of the edge w.r.t. the source of the edge, so, for edge e the attribute $attr(e)$ is given by $attr(e) = position(target(e)) - position(source(e))$.
2. Calculate SDA and SDB. Copy the edge attributes described in step 1 to the corresponding edges of SDA and SDB.
3. This is the crucial step. First compute the overlay SDC. As described in method3, during the overlay construction, every edge e of SDC gets two attributes indicating from which edge of SDA or SDB e stems. Using these attributes, calculate the attributes for every edge e of SDC in the following way. When e stems from only one edge, so an edge of SDA (exclusive)-or SDB, e is given the vector attribute of this edge. When e stems from an edge of SDA and an edge of SDB, e gets as attribute the vector sum of the attributes of these edges.
4. Calculate the dual of SDC, called C. Of every edge of SDC the edge attribute is copied to the corresponding edge of C.
5. Calculate node attributes of C, representing vertex positions of C as follows. Choose some node n_0 of C and assign to it some freely chosen position $pos(n_0)$, for example $(0, 0, 0)$. Then for every edge e which has n_0 as source, visit the node target(e), and assign to it the attribute $pos(n_0) + attr(e)$, i.e. to all nodes directly connected with n_0 are assigned the position of n_0 plus the edge vector of the connecting edge. This process is continued until every node has been visited.
6. Shift C to the correct position. This is done with three similar operations, one for the x direction, one for the y direction and one for the z direction. We explain the x direction shift.
   Let A_max_x be the x position of the most extreme point(s) of A in the positive x direction, and similarly for B_max_x and C_max_x. It can be easily checked that it should hold that

$$C_max_x = A_max_x + B_max_x. \tag{2}$$

In the previous step C was placed at a provisional position in space. Let $prov_C_max_x$ be the maximal x position of C at this provisional position. Then, by shifting C over

$$A_max_x + B_max_x - prov_C_max_x \tag{3}$$

C gets its correct position in the x direction. By a similar shift in the y and z direction C gets its correct position.

6 Discussion

Method3 works because a convex polyhedron is defined by its vertices. Method4 works because a polyhedron is defined, up to its absolute position, by its edge vectors. Method4 works without face location. Instead, edge location is done. The advantage of method4 over method3 is that edge location has already been done during the overlay phase without overhead. In terms of the number of computations, the advantage of not having to do face location outweighs the need to restore the absolute position of C.

As mentioned before, the overlay may be computed in time $O(ne_A+ne_B+k)$. The time complexity of computing SDA and SDB, of attribute manipulation, and of shifting C to the correct position, is inferior to the time complexity of computing the overlay, so, method4 has a time complexity of $O(ne_A+ne_B+k)$. Obviously, this is better than method1 and method2. Probably it is also better than method3 because we think that the time complexity of face location is greater than $O(ne_A + ne_B + k)$.

Why does method4 work? We can prove that it is correct by using the support function [9, 10], but because of limited space it can not be given here. We only make a few remarks which may serve as a starting point of a proof.

As remarked earlier, most faces of C are identical to the faces of A and B, plus additional paralellogram faces with one edge from A and one edge from B. So, the edges of C only consist of edges of A and B, and occasionally an edge that is the sum of an edge of A and an edge of B. The last type of edge occurs when an edge in SDA (partially) coincides with an edge in SDB. In step 3 of method4 these three kinds of edges of C are created, that is, every edge of C is an edge of A or of B or the sum of an edge of A and of B.

Another remark. The edges of SDC (partially) coincide with edges of SDA or SDB. To cover completely an edge e of say SDB with edges of SDC may require two or more edges of SDC. In step 3 of method4 each of these edges of SDC gets the same attribute, indicating that C gets some parallel edge vectors. More precisely: When an edge of SDB is subdivided in SDC in n edges this indicates that C will have n parallel edges, parallel to the corresponding edge of B. See fig. 3.

7 Conclusion

We have shown that the Minkowski sum of two convex polyhedra may be computed almost entirely in the slope diagram domain, and that the usual face

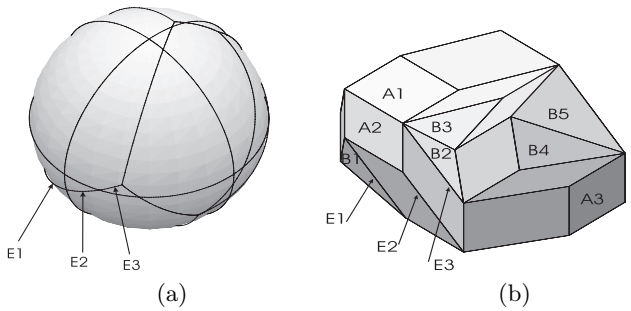


Fig. 3. The slope diagram SDC and C from figure 1. In SDC an edge of SDB is subdivided in three edges of SDC (See three arrows $E1..E3$). This results in three parallel edges in C (See three arrows $E1..E3$) between $B1$ and $B2$. In B (see figure 1) there was only one edge between $B1$ and $B2$. In C the faces from A and B are marked with $A1..A3$ and $B1..B5$.

location can be avoided, leading to a more efficient algorithm. The crucial part of the method is the construction of the attributed overlay of the slope diagrams of A and B. Further, only simple attribute manipulations and simple geometrical computations are used. The time complexity of the method is linear in the size of the input plus output.

Literature

[1] H. Bekker, J. B. T. M. Roerdink: Calculating critical orientations of polyhedra for similarity measure evaluation. Proc. of the IASTED Int. Conf. Computer Graphics and Imaging, 1999, Palm Springs, USA. p. 106-111
[2] A. V. Tuzikov, J. B. T. M. Roerdink, H. J. A. M. Heijmans: Similarity Measures for Convex Polyhedra Based on Minkowski Addition. Pattern Recognition 33 (2000) 979-995
[3] L. J. Guibas, R. Seidel: Computing convolutions by reciprocal search. Discrete and Computational Geometry. Vol. 2, p. 175-193, 1987.
[4] M. de Berg, M. van Kreveld, M. Overmars, O. Schwarzkopf: Computational Geometry, Algorithms and Applications. Springer Verlag, Berlin. (1997)
[5] P. K. Ghosh: A unified computational framework for Minkowski operations. Comput. & Graphics, Vol. 17, No. 4, 1993.
[6] U. Finke, K. H. Hinrichs: Overlaying simply connected planar subdivisions in linear time. Proc. of the 11th Int. symposium on computational geometry, 1995.
[7] A. M. Brinkmann: Entwicklung und robuste Implementierung eines laufzeitoptimalen Verschneidungsoperators für Trapezoidzerlegungen von thematishen Karten. PhD. Thesis, University of Münster, 1998.
[8] K. Melhorn, S. Näher: LEDA A Platform for Combinatorial and Geometric Computing. Cambridge University press,Cambridge. 1999
[9] H. G. Eggleston: Convexity. Cambridge University Press, Cambridge. 1958
[10] H. Busemann: Convex Surfaces. Interscience, Inc., New York. 1958

REGTET: A Program for Computing Regular Tetrahedralizations

Javier Bernal

National Institue of Standards and Technology, Gaithersburg MD 20899, USA,
javier.bernal@nist.gov,
WWW home page: http://math.nist.gov/~JBernal

Abstract. REGTET, a Fortran 77 program for computing a regular tetrahedralization for a finite set of weighted points in 3−dimensional space, is discussed. REGTET is based on an algorithm by Edelsbrunner and Shah for constructing regular tetrahedralizations with incremental topological flipping. At the start of the execution of REGTET a regular tetrahedralization for the vertices of an artificial cube that contains the weighted points is constructed. Throughout the execution the vertices of this cube are treated in the proper lexicographical manner so that the final tetrahedralization is correct.

1 Introduction

Let S be a finite set of points in 3−dimensional space ($\mathcal{R}^3$). By a *tetrahedralization* T for S we mean a a finite collection of tetrahedra (3-dimensional triangles) with vertices in S, that satisfies the following two conditions.

1. Two distinct tetrahedra in T that are not disjoint, intersect at a common facet, a common edge, or a common vertex.
2. The union of the tetrahedra in T equals the convex hull of S.

For each point p in S let w_p be a real-valued weight assigned to p. Given p in S and a point x in $\mathcal{R}^3$, the *power distance* of x from p, denoted by $\pi_p(x)$, is defined by

$$\pi_p(x) \equiv |xp|^2 - w_p,$$

where $|xp|$ is the Euclidean distance between x and p. Given a tetrahedron t with vertices in S, a point, denoted by $z(t)$, exists in $\mathcal{R}^3$ with the same power distance, denoted by $w(t)$, from all vertices of t. Point $z(t)$ is called the *orthogonal center* of t. Given a tetrahedralization T for S, we then say that T is a *regular tetrahedralization* for S if for each tetrahedron t in T and each point p in S, $\pi_p(z(t)) \geq w(t)$. We observe that T is unique if for each tetrahedron t in T and each point p in S that is not a vertex of t, $\pi_p(z(t)) > w(t)$. If T is unique then the *power diagram* of S [1] is the dual of T. Finally, we observe that if the weights of the points in S are all equal then the power diagram of S is identical to the *Voronoi diagram* of S [10], and the regular and *Delaunay* [4] tetrahedralizations for S coincide.

V.N. Alexandrov et al. (Eds.): ICCS 2001, LNCS 2073, pp. 629–632, 2001.
© Springer-Verlag Berlin Heidelberg 2001

In this paper we discuss REGTET, a Fortran 77 program for computing regular tetrahedralizations (or Delaunay tetrahedralizations in the absence of weights) with incremental topological flipping [6] and lexicographical manipulations [3]. A copy of program REGTET that includes instructions for its execution can be obtained from `http://math.nist.gov/~JBernal`

2 Incremental Topological Flipping

Let T be a tetrahedralization for S, let t be a tetrahedron in T, and let p be a point in S that is not a vertex of t. Denote the vertices of t by q_1, q_2, q_3, q_4, and let T_1 and T_2 be the only two possible tetrahedralizations for $\{q_1, q_2, q_3, q_4, p\}$ [9]. Assume t is in T_1, and T_1 is contained in T. A *topological flip* or simply a *flip* on T_1 is an operation that replaces T_1 with T_2 in T.

Program REGTET which is based on an algorithm by Edelsbrunner and Shah [6] constructs a regular tetrahedralization for the set S by adding the points in S one at a time into a regular tetrahedralization for the set of previously added points. A point is added by REGTET through a finite number of steps, each step involving a decision about whether a certain flip should take place and if so applying the flip. This technique is a generalization of a result for computing incrementally Delaunay triangulations in $\mathcal{R}^2$ [7]. By extending results for Delaunay triangulations and tetrahedralizations [8], [9], Edelsbrunner and Shah [6] justify their algorithm.

3 Lexicographical Manipulations

The incremental nature of Edelsbrunner and Shah's algorithm [6] implies that before any points in S are added a regular tetrahedralization must be first constructed by program REGTET with vertices close to infinity and underlying space equal to $\mathcal{R}^3$. The vertices of this initial tetrahedralization are said to be *artificial*. Throughout the execution of the program artificial points must be treated in the proper lexicographical manner so that the final tetrahedralization does contain a tetrahedralization for S, and this tetrahedralization for S is indeed regular (since the coordinates of the artificial points can be extremely large in absolute value, it is inadvisable to identify them, thus the need to treat artificial points in a lexicographical manner).

Lexicographical manipulations that are employed in program REGTET are described and justified in [3]. At the start of the execution of the implementation a $3-$dimensional cube with vertices close to infinity that contains S in its interior is identified, and a regular tetrahedralization for the set of vertices of the cube (weights set to the same number) is computed. The execution then proceeds with the incremental insertion of points in S as suggested by Edelsbrunner and Shah. However, at all times, because of the lexicographical manipulations employed in the presence of artificial points (the vertices of the cube), the artificial points are assumed to be as close to infinity as the manipulations require.

4 Flipping History

At all times during its execution, program REGTET maintains a list of all tetrahedra in the current and previous tetrahedralizations. This list is in the form of a directed acyclic graph that represents the history of the flips REGTET has performed [6], and it is used by REGTET for identifying a tetrahedron in the current tetrahedralization that contains a new point. Identifying a tetrahedron that contains a point this way is a generalization of a technique used in [7] for 2−dimensional triangulations.

5 Running Time

Program REGTET has the capability of adding the points in S in a random sequence. For some positive integer n, let n be number of points in S. Using an analysis similar to the one in [7] for 2−dimensional Delaunay triangulations, Edelsbrunner and Shah [6] show that if the points in S are added in a random sequence then the expected running time of their algorithm for computing a regular tetrahedralization for S is $O(n \log n + n^2)$. As pointed out in [6], the actual expected time could be much less, i. e. the second term (n^2) in the above expectation could be much less, depending on the distribution of the points in S. Accordingly this should be the case for sets of uniformly distributed points in a cube or a sphere. As proven for a cube in [2] and for a sphere in [5], the complexity of the Voronoi diagram, and therefore of the Delaunay tetrahedralization, for such sets is expected linear. Indeed we have obtained good running times when computing with REGTET regular tetrahedralizations for sets of uniformly distributed points in cubes: on the SGI ONYX2 (300 Mhz R12000 CPU) the running time is about 25 CPU minutes for a set of 512,000 points with random weights. A similar time was obtained for the same set without weights. Finally, REGTET has also been executed successfully and efficiently to compute Delaunay tetrahedralizations for non-uniformly distributed point sets representing sea floors and cave walls.

References

1. Aurenhammer, F.: Power diagrams: properties, algorithms and applications. SIAM J. Comput. **16** (1987) 78–96
2. Bernal, J.: On the expected complexity of the 3−dimensional Voronoi diagram. NISTIR 4321 (1990)
3. Bernal, J.: Lexicographical manipulations for correctly computing regular tetrahedralizations with incremental topological flipping. NISTIR 6335 (1999)
4. Delaunay, B.: Sur la sphère vide. Bull. Acad. Sci. USSR (VII), Classe Sci. Mat. Nat. (1934) 793–800
5. Dwyer, R. A.: Higher-dimensional Voronoi diagrams in linear expected time. Discrete Comput. Geom. **6** (1991) 343–367
6. Edelsbrunner, H., Shah, N. R.: Incremental topological flipping works for regular triangulations. Algorithmica **15**(3) (1996) 223–241

7. Guibas, L. J., Knuth, D. E., Sharir, M.: Randomized incremental construction of Delaunay and Voronoi diagrams. Springer-Verlag Lecture Notes in Computer Science **443** (1990) 414–431
8. Lawson, C. L.: Software for C^1 surface interpolation. Mathematical Software III, J. R. Rice (Ed.), Academic Press, New York (1977) 161–194
9. Lawson, C. L.: Properties of n-dimensional triangulations. Computer Aided Geometric Design **3** (1986) 231–246
10. Voronoi, G.: Nouvelles applications des paramètres continus à la théorie des formes quadratiques. J. Reine Angew. Math. **134** (1908) 198–287

Fast Maintenance of Rectilinear Centers

Sergei Bespamyatnikh[1] and Michael Segal[2]

[1] Department of Computer Science, University of British Columbia,
Vancouver V6T 1Z4, Canada
besp@cs.ubc.ca, http://www.cs.ubc.ca/spider/besp
[2] Department of Communication Systems Engineering,
Ben-Gurion University of the Negev, Beer-Sheva 84105, Israel
segal@cs.bgu.ac.il, http://www.cs.bgu.ac.il/~segal

Abstract. We address the problem of dynamic maintenance of 2-centers in the plane under rectilinear metric. We present two algorithms for the continuous and discrete versions of the problem. We show that rectilinear 2-centers can be maintained in $O(\log^2 n)$ time. We give an algorithm for semi-dynamic (either insertions only or deletions only) maintenance of the discrete 2-centers in $O(\log n \log m)$ amortized time where n is the number of customer points and m is the number of possible locations of centers.

1 Introduction

Given two sets S, C of points in the plane of size n and m, respectively we wish to maintain dynamically (under insertions and/or deletions of points of S)

1. **Rectilinear 2-center:** two squares that cover S such that the radius of maximal square is minimized.
2. **Discrete Rectilinear 2-center:** two squares that cover S centered at points of C such that the radius of maximal square is minimized.

We also consider the generalization of problem 2 for the case of rectangles, where one wants to minimize the largest perimeter. There are several results for the static version of the problems above. A linear time algorithm for the planar rectilinear 2-center problem is given by Drezner [4]. The $O(n \log n)$ time solution for the discrete rectilinear 2-center was given by Bespamyatnikh and Segal [3] and the optimality of their algorithm has been shown by Segal [6]. To our best knowledge nothing has been done regarding the dynamic version of the rectilinear 2-center problem. Bespamyatnikh and Segal [3] considered also a dynamic version of the discrete rectilinear 2-center. They have been able to achieve an $O(\log n)$ update time, though the actual query time is only $O(m \log n(\log n + \log m))$.

For the dynamic rectilinear 2-center problem we present a scheme which allows us to maintain an optimal solution under insertions and deletions of points of S in $O(\log^2 n)$ time (both update and query), after $O(n \log n)$ preprocessing time. For the semi-dynamic discrete rectilinear 2-center problem we give an algorithm for maintaining the optimal pair of squares under insertions only (resp. deletions only) of points of S in amortized $O(\log n \log m)$ time (both update and query), after $O(n \log n)$ preprocessing time. Our solution for the semi-dynamic

V.N. Alexandrov et al. (Eds.): ICCS 2001, LNCS 2073, pp. 633–639, 2001.
© Springer-Verlag Berlin Heidelberg 2001

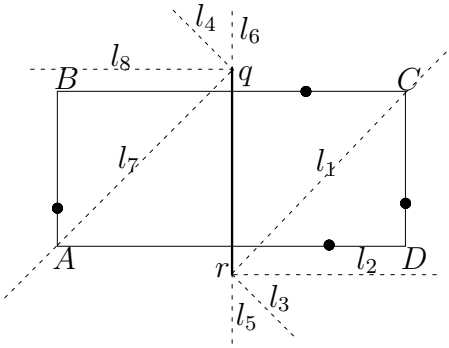


Fig. 1. Subdivision of the bounding box into the ranges.

discrete rectilinear 2-center improves the best previous result by almost linear factor, thus providing first sublinear semi-dynamic algorithm for dynamic maintenance of the discrete rectilinear 2-center.

2 Dynamic Rectilinear 2-Center

Denote by $|pq|$ the L_∞ distance between two points p, q in the plane. We observe as in [2] that two pairs of the diagonal vertices of the bounding box of S play a crucial role in defining two minimal squares that cover S. More precisely, let us consider a pair of diagonal vertices A and C of the bounding box of S in Figure 1. For the vertex A we find the farthest neighbor point $p' \in S$ (in L_∞ metric) among the points that are closer to A than to C. We repeat the similar procedure for vertex C, obtaining point p''. It can be done efficiently by constructing a rectilinear bisector l_4qrl_3 and dividing the obtained regions into the wedges, see Figure 1. The main property of such subdivision is that the largest distance from a point $p_i \in W$ (W is a wedge) to corresponding vertex (A or C) is either x- or y-distance between p_i and the corresponding vertex. For example, consider the diagonal vertex C in Figure 1 and associated with C wedges: l_4ql_6, l_6qrl_1, l_1rl_2, l_2rl_3 (we should consider all these wedges since it may happen that points q and r will be inside of the bounding box of S). We can use the orthogonal range tree data structure [1] in order to find the required largest distance. For the case of wedge l_1rl_2, only the y-coordinate of any point of S lying in this wedge determines the distance from this point to C. We construct a range tree T in the new system of coordinates corresponding to the directions of l_1 and l_2. The main structure of T is a balanced binary tree according to the "x"-coordinate of points. Each node v of this tree corresponds to the balanced binary tree (secondary tree) according to the "y"-coordinate of points whose "x"-coordinate belongs to the subtree rooted at v. We augment this data structure by keeping an additional value for each node w in the secondary data structures as the minimal value of the actual x-coordinates of the points corresponding to

the nodes in the subtree rooted at w. In order to find the farthest l_∞ neighbor of C in the wedge l_1pl_2, we perform a query on t by taking this wedge as a range. At most $O(\log^2 n)$ nodes of the secondary data structure are taken into account and we collect all the minimal x-values that are kept in these nodes. A point that has a minimal x-coordinate is a farthest neighbor of C in the wedge l_1pl_2.

We apply the similar technique for the remaining wedges. The entire update and query procedure takes $O(\log^2 n)$ time after initial $O(n \log n)$ time for the construction of the orthogonal range trees. In this way we can compute points p' and p''. Let δ_1 be the maximal value between $|Ap'|$ and $|Cp''|$. Using the same searching farthest neighbor technique for a different pair of diagonal vertices B and D, we obtain points $q', q'' \in S$ such that $|Bq'| = \max_{q \in S, |Bq| \leq |Dq|} |Bq|$ and $|Dq''| = \max_{q \in S, |Dq| < |Bq|} |Dq|$. Let $\delta_2 = \max(|Bq'|, |Dq''|)$. Finally, the smallest value between δ_1 and δ_2 defines the size of the squares and their position in the optimal solution of the rectilinear 2-center problem.

3 Dynamic Discrete Rectilinear 2-Center

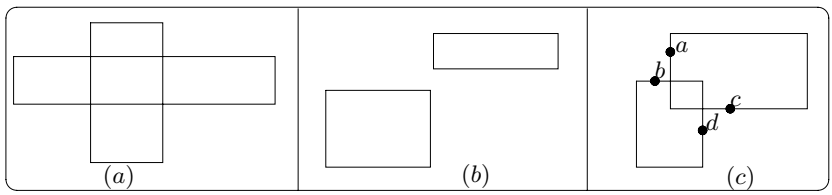


Fig. 2. Different configurations of bounding boxes defined by two optimal discrete squares.

First, we consider an optimal solution for the static discrete rectilinear 2-center problem. Let s_1 and s_2 be two optimal discrete squares centered at points of C that cover S. Consider the bounding boxes B_1 and B_2 of points covered by s_1 and s_2, respectively. Three different configurations of B_1 and B_2 are possible, see Figure 2. (In fact, in our analysis a few more different configurations appear, but they are symmetrically opposite to the configurations described below). Denote by $bb(S)$ the bounding box of S. We call a point of S a *determinator* if it lies on one of the edges of $bb(S)$. Normally, $bb(S)$ has four determinator points $r, l, t, b \in S$ that lie onto the right, left, top and bottom sides of $bb(S)$, respectively. Configuration (a) is characterized by fact that each one of the bounding boxes B_1 and B_2 has two opposite determinators on its sides, e.g., r and l lie on the edges of B_1 while b and t lie on the edges of B_2. In configurations (b) and (c), each one of the bounding boxes B_1 and B_2 has two adjacent determinators on its sides, e.g., l and b lie on the edges of B_1 while r and t lie on the edges of

B_2. The main difference between these two configurations is that in case (b) B_1 and B_2 are totally disjoint, while in case (c) B_1 and B_2 intersect.

For each configuration we find an optimal pair of discrete squares as follows. First, we consider case (a). We show that one of the squares s_1 and s_2 contains three determinators. Without loss of generality we assume that the width of the bounding box of S is greater or equal to its height. Suppose that B_1 contains left and right determinators. Then s_1 contains either upper or lower (or both) determinator. Therefore we may assume that B_1 and B_2 are totally disjoint; moreover, one of them contains three determinators. This case can be solved easily by applying a binary search on the sorted list of x-coordinates (y-coordinates) of the points of S. Each step of the binary search splits the points of S into two subsets $S_1, S_2 \subset S$. For each subset S_i, we compute its bounding box $B_i, i = 1, 2$ (we can assume that one of the bounding boxes contains three determinators).

Now, we need to find two smallest discrete squares s_1 and s_2 that cover B_1 and B_2, respectively. Consider the bounding box B_1 and its center c_1. Without loss of generality the width of B_1 is greater or equal to its height. Our goal is to find the closest L_∞ neighbor point $q \in C$ to the vertical segment A_1A_2 (note that that is defined as follows. A_1A_2 passes through center c_1, the ray emanating from left-bottom corner of B_1 in direction to A_2 makes $45°$, and A_1 lies on $(-45°)$-ray from left-top corner of B_1, (see Figure 3). This point q will define the center of the discrete square s_1. We can find q using orthogonal range trees by the similar technique described in the previous section. We divide the search region into the wedges as shown in the Figure 3, such that the smallest distance from a point $q_i \in W$ (W is a wedge) to A_1A_2 is either x- or y-distance (depending on the wedge) between q_i and A_1A_2.

After we found the locations and sizes of s_1 and s_2 we guide a binary search in order to get an optimal size for the squares for this configuration. Notice that the configuration (b) can be solved by the same method by applying two binary searches on the points (according to x- and y-coordinates) of S. In each step of a binary search we obtain disjoint boxes B_1 and B_2. We find a minimal discrete square that covers B_i, $i = 1, 2$ using orthogonal range trees. The total time required for case(b) (and case(a)) is $O(\log n \log m)$.

The case (c) is most interesting and it can be solved using the following approach. The bounding boxes B_1 and B_2 form two orthogonal corners with four points $a, b, c, d \in S$, see Figure 2(c). The additional property is that $B_1 \cap B_2 \neq \emptyset$. We conclude that the points a, b form a single link in the upper-left staircase chain of the points of S and the points c, d form a single link in the lower-right staircase chain of the points of S. These two chains correspond the maximal upper-left (north-west) and lower-right (south-east) points of S (similar to set of maxima of S and set of minima of S, Chapter 4 [5]). Each pair of corners: one from the upper-left staircase and one from the lower-right staircase define a configuration with two discrete squares that cover S. For each corner on the upper-left staircase we find the best corresponding corner (in terms of the largest

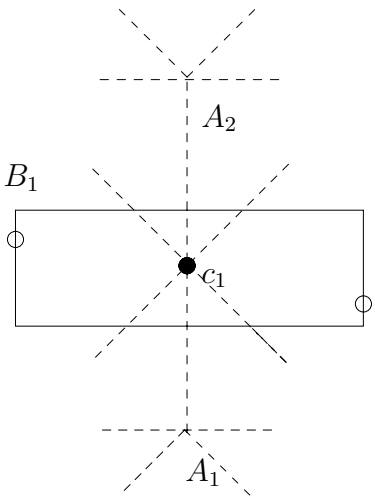


Fig. 3. Regions for point $q \in C$.

size of two obtained squares) on the lower-right staircase and put a *pointer* between them, see Figure 4.

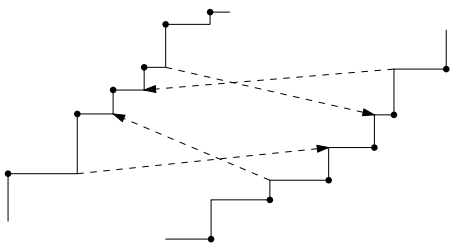


Fig. 4. Pointers between staircases.

We perform the similar operation for the corners in the lower-right staircase. Thus, we have a collection of at most $2n$ pointers. It may happen that two or more pointers refer to the same corner; in this case we store only one pointer that defines the best two discrete squares. In fact, we keep the sizes of squares in the heap (as an appropriate pointer with associated size of square). Notice that, for a particular corner c as a source, we can find its pointer in $O(\log n \log m)$ using a binary search with orthogonal range trees. For fixed c, the bounding box B_1 (and B_2) has three fixed sides. B_1 changes monotonically when we traverse corners on the opposite staircase. Therefore the size of the discrete square covering B_1 changes monotonically and we can apply binary search. For any corner on the

opposite staircase, the discrete square containing B_1 can be obtained in $O(\log m)$ time using range trees. The binary search finds two corners c' and c'' such that corresponding sizes s'_1, s'_2 and s''_1, s''_2 of the squares satisfy the following property: $s'_1 \leq s'_2$ and $s''_1 \geq s''_2$. The total time for finding a pointer is $O(\log n \log m)$.

Consider the insertion of a new customer point p. If p lies between two staircases, then it does not make any change to the staircases and our current solution (case (c)). Suppose that p is above left staircase (the case of right staircase is symmetric), see Figure 5. First, we update the staircase. We find the sequence of corners that are no longer valid. Otherwise, we update a corresponding staircase, remove non-valid pointers and compute two new pointers from the corners defined by the new inserted point and its neighbors in the staircase. If only insertions are allowed (or deletions) the total number of changes in the staircases is $O(n)$ and, therefore, we achieve an amortized $O(\log n \log m)$ time for updates and queries.

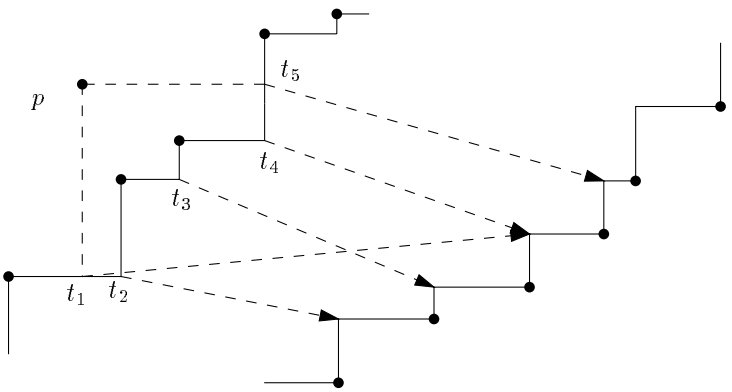


Fig. 5. Insertion of point p. Corners t_2, t_3, t_4 and their pointers are deleted and two corners t_1 and t_5 with pointers are inserted.

Theorem 1. *Rectilinear 2-centers can be maintained in amortized* $O(\log n \log m)$ *time in semi-dynamic data structure of linear size.*

4 Future Work

In the extended version of this paper we also show how to maintain an $(1+\varepsilon)$-approximated solution for the discrete two-center problem in $O\left(\frac{1}{\varepsilon}\log(n+m)\right)$ time by supporting both deletions and insertions. We also show how to solve efficiently the discrete two-center rectangular problem. A possible future directions for research are containing the extension of the results obtained in this paper to higher dimensions, making algorithms fully dynamic and considering the Euclidean metric.

References

1. M. de Berg, M. van Kreveld, M. Overmars, O. Schwartzkopf *Computational Geometry, Algorithms and Applications*, Springer-Verlag, 1997.
2. S. Bespamyatnikh and D. Kirkpatrick, "Rectilinear 2-center problems", in *Proc. of 11th Can. Conf. Comp. Geom.*, pp. 68–71, 1999.
3. S. Bespamyatnikh and M. Segal, "Rectilinear static and dynamic discrete 2-center problems", in *Int. Jour. of Math. Algorithms*, to appear.
4. Z. Drezner, "On the rectangular p-center problem", *Naval Res. Logist. Q.*, 34, pp. 229–234, 1987.
5. F. P. Preparata and M. I. Shamos, "Computational Geometry: An Introduction", *Springer-Verlag*, 1990.
6. M. Segal, "Lower bounds for covering problems", manuscript, 1999.

Exploring an Unknown Polygonal Environment with Bounded Visibility

Amitava Bhattacharya[1], Subir Kumar Ghosh[1], and Sudeep Sarkar[2]

[1] School of Technology and Computer Science, Tata Institute of Fundamental Research, Mumbai 400005, India
email: {amitava@tcs.tifr.res.in, ghosh@tifr.res.in}
[2] *** Department of Computer Science and Engineering, University of South Florida, Tampa, USA 33620
email:{sarkar@csee.usf.edu}

Abstract

This paper integrates constraints from visual processing and robot navigation into the well-studied computational geometry problem of exploring unknown polygonal environments with obstacles. In particular, we impose two constraints. First, we consider a robot with limited visibility, which can reliably "see" its environment only within a certain range R. Second, we allow the computation of visibility only from discrete number of points on the path. Both these constraints arise from real life cost constraints associated with robotic exploration and robot vision processing. We present an online algorithm for exploration under such constraints and show that the maximum number of views needed to explore the unknown polygon (with obstacles) P, is bounded by $\left\lfloor \frac{2\times \text{Perimeter}(P)}{R} \right\rfloor + \left\lfloor \frac{2\times \text{Area}(P)}{R^2} \right\rfloor + 3n$, where n is the number of sides of the polygon. We also show that the competitive ratio of the algorithm is $6\pi + \frac{4r+3n}{max(\left\lceil \frac{\text{Area}(P)}{\pi R^2} \right\rceil, \left\lceil \frac{\text{Perimeter}_{(P)\,-\,2Rr}}{2\pi R} \right\rceil)}$.

Keywords: Bounded visibility, Discrete visibility, Polygonal environment, Exploration, On-line algorithm

1 Introduction

The context of this work is the exploration of *unknown* polygonal environments *with* obstacles. Both the outer boundary and the inside obstacles boundaries are piecewise linear. The boundaries can be non-convex. Can we construct a complete representation of the environment, in terms of say its triangulation, as we explore it incrementally starting from a given position? Most online computational geometry algorithms for exploring unknown polygons [1,2,4,5,6,7,14,15, 19,20,25], assume that the visibility region can be determined in a continuous fashion from each point on a path and that we have infinite visibility. While such

*** This work was done when the author visited Tata Institute of Fundamental Research on sabbatical leave from the University of South Florida, Tampa.

V.N. Alexandrov et al. (Eds.): ICCS 2001, LNCS 2073, pp. 640–648, 2001.
© Springer-Verlag Berlin Heidelberg 2001

assumptions are reasonable in the case of a human "watchman," they are not practical in the context of real-life robot navigation. First, autonomous robots can only carry a limited amount of on-board computing capability. At the current state of the art, computer vision algorithms for computing visibility polygons are time consuming [3,9,16,21]. The computing limitations dictate that it is not practically feasible to continuously compute visibility polygons along the robot's trajectory. Second, for good visibility, the robot's camera will be typically mounted on a mast. Such devices vibrate during movement, and hence for good precision (which is required to compute an accurate visibility polygon) the camera must be stationary for each view. Third, computer vision range sensors or algorithms, such as stereo or structured light range finder, can reliably compute the 3D scene locations only up to a depth R. The reliability of depth estimates is inversely related to the distance from the camera. Thus, the range measurements from a vision sensor for objects that are far away are not at all reliable. This suggests that it is necessary to modify exploration algorithms to make them more realistic by restricting visibility polygons by the range distance R. Therefore, the portion of the boundary of a polygonal environment within the range distance R is only considered to be visible from the camera of the robot. We refer to the visibility polygon under this range restriction as the *restricted visibility polygon*. Observe that restricted visibility polygons need not be always a closed boundary. In fact, it can consists of several disjoint polygons chains. So, exploring an unknown polygonal environment using restricted visibility polygons requires more number of views. In earlier works, Ghosh and Burdick [13,11,12] have presented algorithms that take into account the first two constraints. In this paper, we generalize their algorithm to take into account the constraint of bounded visibility and analyze its computational consequences.

The essential components that contribute to the total cost, in terms of time, required for a robotic exploration can be analyzed as follows [13]. Each move will have two associated costs. First, there is the time required to physically execute the move. If we crudely assume that the robot moves at a constant rate, r, during a move, the total time required for motion will be $\frac{d}{r}$, where d is the total path length followed by the robot during the exploration. Second, there are the exploratory processes in which the robot plans its moves based on its most recent geometric information about the scene. This cost has two subcomponents: time spent by the on board sensors to acquire information about the scene and the time spent to plan its next move. Let the average times spent on acquiring sensory information and planning be t_S and t_M, respectively, per operation. Let N_M and N_S be respectively the number of moves and the number of sensor operations that are required to complete the exploration of P. Then the total time, T, required to explore is

$$T(P) = t_M \ N_M + t_S \ N_S + \frac{d}{r} \tag{1}$$

In practice, the first two terms ($t_M \ N_M + t_S \ N_S$) is indeed a significant fraction of $T(P)$. Thus, we would like any exploration algorithm to minimize N_M and N_S. In this paper, we present an on-line algorithm that requires at most

$\left\lfloor \frac{2\times \text{Perimeter}(P)}{R} \right\rfloor + \left\lfloor \frac{2\times \text{Area}(P)}{R^2} \right\rfloor + 3n$ views for exploring an unknown polygonal environment.

In the next section, we review the relevant prior work. In Section 3 we present the Ghosh and Burdick algorithm, which needs only discrete set of visibility computations but does not handle bounded visibility. In Section 4 we present a generalized version of the exploration algorithm for bounded visibility in and prove its correctness. We also establish an upper bound on the number of views required to explore the entire region. In Section 5 we conclude the paper with a few remarks.

2 Prior Work

There have been many algorithms proposed for robot navigation in unknown environments. Rao [23] considered problem of navigating *generalized* polygons that have both straight and curved edges. Taylor and Kriegman [26] present landmark based exploration algorithms that are heavily dependent on vision sensors. Range sensor based navigation algorithm include that proposed by Foux *et al.* [10], Kamon *et al.* [17,16], and Ekman *et al.* [8]. The exploration strategy of Ekman *et al.* [8] is closest to that proven to be optimal by Ghosh and Burdick [13, 11,12]. Ekman *et al.* assumes a point robot with an ideal range sensor that can see up to infinity and can measure range in N uniformly distributed directions. Thus, the range sensors offer a sampled version of the visibility polygon. The edges of the visibility polygon that do not correspond to environmental edges are termed as the jump edges, which are used to suggest the next exploration point. Ghosh and Burdick proves that such a strategy is indeed an optimal one and is guaranteed to completely explore the polygonal environment.

The consideration of bounded visibility in computational geometry algorithms is rare. We could locate just one work that also uses the notion of bounded visibility but in a different context. Kim *et al.* [18] coined the notion of d-visibility to describe the situation when a robot as restricted range of sight. They presented optimal algorithms to find the edge visibility polygon under the d-visibility constraint.

3 Ghosh-Burdick Algorithm

In this section, we summarize the exploration algorithm of Ghosh and Burdick for a point robot. The robot starts at any initial location, p_1, where it determines the visibility polygon $\mathcal{V}(P, p_1)$. Using the visible vertices of P in $\mathcal{V}(P, p_1)$, the robot triangulates as much of the $\mathcal{V}(P, p_1)$ as possible. Let this triangulation be denoted $T(P)$. The robot then executes a forward move to another position $p_2 \in \mathcal{V}(P, p_1)$ and computes the next visibility polygon $\mathcal{V}(P, p_2)$. The region common to $\mathcal{V}(P, p_2)$ and $T(P)$ is removed. The remaining free space in $\mathcal{V}(P, p_2)$ is triangulated and added to $T(P)$. This process is repeated till the entire free space is explored. It may appear that to see the entire free space it is enough

to see all the vertices and edges of P. But it is not the case as shown by the example in Figure 1.

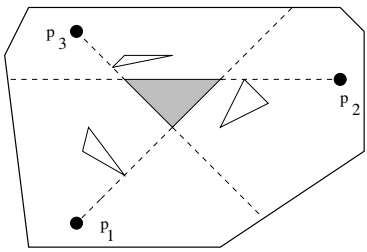


Fig. 1. Three views are enough to see all vertices and edges but not the entire free space.

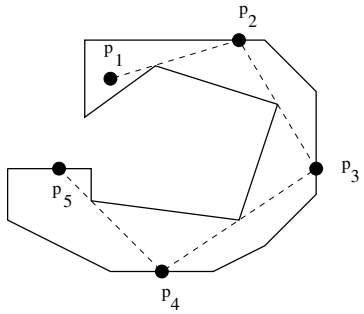


Fig. 2. The spiral polygon can be explored in r+1 steps.

In the following, we present the major steps of their algorithm.

Step 1: $i := 1$; $T(P) := \emptyset$; $S := \emptyset$; Let p_1 denote the starting position of the robot.

Step 2: Compute $\mathcal{V}(P, p_i)$; $T(P) := T(P) \cup T'(P)$ where $T'(P)$ is the triangulation of $\mathcal{V}(P, p_i)$; $S = S \cup p_i$;

Step 3: While $\mathcal{V}(P, p_i) - T(P) = \emptyset$ and $i \not= 0$ then$i := i - 1$;

Step 4: If $i = 0$ then goto Step 7;

Step 5: If $\mathcal{V}(P, p_i) - T(P) \not= \emptyset$ then choose a point z on any constructed of $\mathcal{V}(P, p_i)$ lying outside $T(P)$;

Step 6: $i := i + 1$; $p_i := z$; goto Step 2;

Step 7: Stop.

Ghosh and Burdick have shown that there exists at least a triangle every time $T'(P) - T(P)$ is computed and the algorithm computes at most $r+1$ views, where r is the total number of reflex vertices in P. It can be seen that $r + 1$ views are also necessary as in Figures 2 and 3. We have the following theorem.

Theorem 1. *The algorithm of Ghosh-Burdick can be used to explore an unknown polygonal environment using restricted visibility polygons if $R \geq D$ where D is the longest line segment which lies completely inside the polygon.*

Proof: The proof follows from the fact that if $R \geq D$, any visibility polygon computed by the algorithm of Ghosh-Burdick is same as the restricted visibility polygon. □

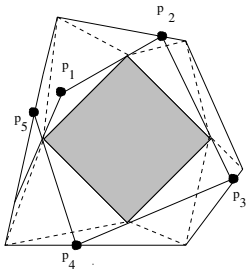


Fig. 3. The polygon is explored in r+1 steps.

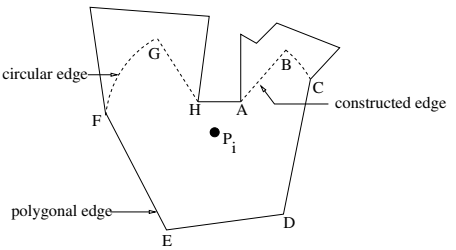


Fig. 4. RVP(p_i)= (ABCDEFGHA).

4 Generalization of Ghosh-Burdick Algorithm

In this section we present our exploration algorithm and establish an upper bound for its competitive ratio. In Theorem 2.1 we have shown that Ghosh-Burdick algorithm can be used for exploring an unknown polygonal environment when $R \geq D$. We now consider the case when $R < D$. Let RVP denote the restricted visibility polygon computed from the same point.

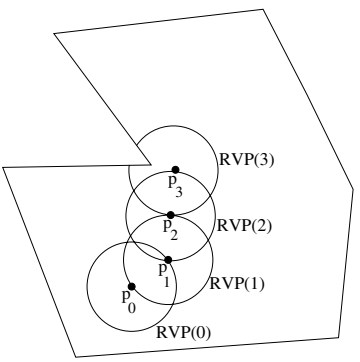


Fig. 5. A sequence of restricted visibility polygons.

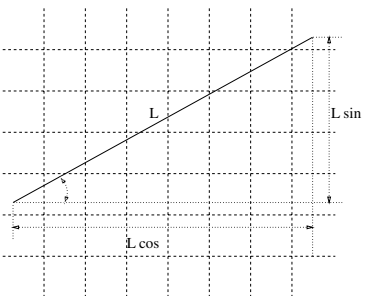


Fig. 6. A line segment of length can cut at most $\left\lfloor \frac{\sqrt{2}L}{l} \right\rfloor + 3$ squares.

Observe that a restricted visibility polygon may not be closed (see Figure 4) and therefore its boundary can have circular edges in addition to constructed and polygonal edges. So it is necessary to take another view from some point in

RVP in order to see more of P. The process can be repeated till the union of these restricted visibility polygons cover P. Let the restricted visibility polygon computed from the point p_i be denoted by $RVP(p_i)$. The boundary of any region B is denoted by $bd(B)$. In the following we present a procedure to explore the polygon starting from a point p_0 using restricted visibility polygons (see Figure 5). The robot initializes its polar co-ordinate system by setting its origin at p_0. Let $CP(p_0)$ denote the region of P so far visible from the robot.

Algorithm $\mathcal{A}$:

Step 1: $CP(p_0) := RVP(p_0)$; if the boundary of $CP(p_0)$ consists of only polygonal edges then goto Step 4.

Step 2: Choose a point p on any circular arc or constructed edges of $CP(p_0)$ and compute $RVP(p)$; $CP(p_0) := CP(p_0) \cup RVP(p)$.

Step 3: If $CP(p_0)$ has circular arc or constructed edges, then goto Step 2.

Step 4: Stop

Let us now prove the correctness of the algorithm. In order to prove its correctness we have to show $CP(p_0) = P$ when the algorithm terminates. It is obvious that $CP(P_0) \subset P$. So to show equality it is enough to show $P \subset CP(p_0)$. Assume on the contrary that $P \not\subset CP(p_0)$. Then there exists a point $p \in P$ and $p \not\in CP(p_0)$. If there exists a path from p to p_0 lying inside $CP(p_0)$, then p belongs to $CP(p_0)$, a contradiction. Otherwise any path from p to p_0 must intersect $bd(CP(p_0))$. Since every edge of $bd(CP(p_0))$ is a polygonal edge, every path from p to p_0 must intersect $bd(P)$ a contradiction. Hence $CP(P_0) = P$ when the algorithm terminates.

In the following lemmas we establish an upper bound for the number of views required by a robot to explore the region P using $\mathcal{A}$.

Lemma 2. *If a line segment of length L lies on a grid of size l, then the number of squares that can be cut on one side is at most* $\left\lfloor \frac{\sqrt{2}L}{l} \right\rfloor + 3$ *(see Figure 6).*

Proof : Let θ be the slope of the line. Then the number of vertical lines it can cut is at most $\left\lfloor \frac{L\cos\theta}{l} \right\rfloor + 1$. Similarly the number of horizontal lines it can cut at most is $\left\lfloor \frac{L\sin\theta}{l} \right\rfloor + 1$. Observe that each time it cuts a vertical line or a horizontal line it enters a new square. Therefore, the maximum number of squares it can cut is $\left\lfloor \frac{L(\sin\theta+\cos\theta)}{l} \right\rfloor + 3$. Note that 1 is added to account for the starting cell. Since the maximum value of $\sin\theta + \cos\theta = \sqrt{2}$, the maximum number of squares it can cut on one side is $\left\lfloor \frac{\sqrt{2}L}{l} \right\rfloor + 3$. □

Lemma 3. *If the area, perimeter and number of vertices of a polygon P are* area(P), perimeter(P) *and n respectively, then the number of views required using algorithm $\mathcal{A}$ is bounded by* $\left\lfloor \frac{2\times\text{Perimeter(P)}}{R} \right\rfloor + \left\lfloor \frac{2\times\text{Area(P)}}{R^2} \right\rfloor + 3n$.

Proof : Suppose there exists a partition of the polygon into cells such that inside each cell the robot can take at most 1 view. Then the number of cells in any such partitions will give an upper bound on the number of views required. Using this approach an upper bound is obtained as follows. The polygon P is

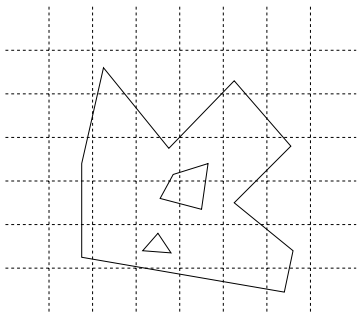


Fig. 7. A polygon P placed on a grid.

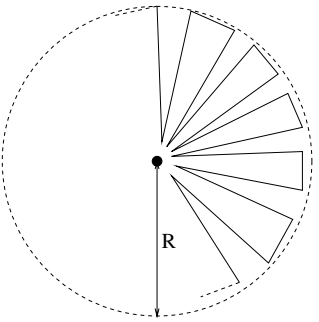


Fig. 8. The polygon can be explored in r+1 views.

placed on a grid where each cell is a square of size $\frac{R}{\sqrt{2}}$ (see Figure 7). Consider the squares that lie totally inside P. Clearly the number of such squares is at most $\frac{2\times Area(P)}{R^2}$. Observe that the robot can take at most one view in each such square since the size of the grid is $\frac{R}{\sqrt{2}}$. So the robot can take at most $\frac{2\times Area(P)}{R^2}$ views from such squares. Let us count the number of views that can be taken from squares which intersect the polygon boundary. Since the grid size is $\frac{R}{\sqrt{2}}$, by Lemma 4.1, a side of length L can cut at most $\lfloor \frac{2.L}{R} + 3 \rfloor$ squares. Consider the squares which are cut by the boundary. The part of these squares which lie inside the polygon can be covered by convex regions with the straight line parts of the polygon boundary as one of its sides. For any such convex region, at most one view can be taken, since the diameter of such regions is less than $\frac{R}{\sqrt{2}}$. So the maximum number of views the robot can take is $\left\lfloor \frac{2\times Perimeter(P)}{R} \right\rfloor + \left\lfloor \frac{2\times Area(P)}{R^2} \right\rfloor + 3n.$ □

Now we derive the competitive ratio for algorithm $\mathcal{A}$. Observe that in each view a robot can see at most πR^2 area of P and $2\pi R + 2Rr'$ of the boundary of P, where r' is the number of reflex vertices seen in that view. Hence even in the best case the robot must take at least $max(\left\lceil \frac{\text{Area}(P)}{\pi R^2} \right\rceil, \left\lceil \frac{\text{Perimeter}(P) - 2Rr}{2\pi R} \right\rceil)$ views. Hence the competitive ratio is

$$\frac{\left\lfloor \frac{2\times \text{Perimeter}(P)}{R} \right\rfloor + \left\lfloor \frac{2\times \text{Area}(P)}{R^2} \right\rfloor + 3.n}{max(\left\lceil \frac{\text{Area}(P)}{\pi R^2} \right\rceil, \left\lceil \frac{\text{Perimeter}(P) - 2Rr}{2\pi R} \right\rceil)}$$

which is upper bounded by $6\pi + \frac{4r+3n}{max(\left\lceil \frac{\text{Area}(P)}{\pi R^2} \right\rceil, \left\lceil \frac{\text{Perimeter}(P) - 2Rr}{2\pi R} \right\rceil)}$.

The worst case arises when the number of reflex vertices is large. Then the number of views required is $r+1$ but two views are sufficient as shown in Figure 8.

5 Concluding Remarks

Observe that when a view is taken from some cell, some portions of other cells may be inside the restricted visibility polygon. Since this fact is not incorporated in calculating the bound, we expect that the number of views will be much less in practice. We also note that as R increases, we need fewer views which is reflected in the bound.

The exploration problem may be considered as a covering P with circles [27] with the additional constraint that between centers a and b of any two circles there is a path composed of line segments $au_1, u_1u_2, ..., u_ib$, such that 1) each of the points $u_1, u_2, ...u_i$ are centers of circles, 2) each segment lies completely in the polygon P and 3) length of each segment is at most R.

Suppose p_1, p_2,...., p_k be the points of P such that an optimal exploration algorithm for a point robot has computed restricted visibility polygons from these points. We know that $\bigcup_{i=1}^{k} RVP(P, p_i) = P$, $p_{i+1} \in V(P, p_i)$ and k is minimum. So, P can be guarded by placing stationary guards at p_1, p_2,...., p_k where it is assumed that each guard can see up to distance R. Hence the exploration problem for a point robot, with limited visibility is the Art Gallery problem with stationary guards [22,24] with the additional constraints. Thus, our exploration algorithm for a point robot is an approximation algorithm for this variation of the Art Gallery problem which also seems to be NP-hard.

References

1. E. Bar-Eli, P. Berman, A. Fiat, and P. Yan. On-line navigation in a room. In *Proceedings of the third ACM-SIAM Symposium on Discrete Algorithms*, pages 237–249, 1992.
2. A. Blum, P. Raghavan, and B. Schieber. Navigating in unfamiliar geometric terrain. In *Proceedings of the 23rd ACM Symposium on Theory of Computing*, pages 494–504, 1991.
3. J. Borenstein, H. R. Everett, and L. Feng. *Navigating mobile robots: sensors and techniques.* A. K. Peters Ltd., Wellesley, MA, 1995.
4. K. Chan and T. W. Lam. An on-line algorithm for navigating in an unknown environment. *International Journal of Computational Geometry and Applications*, 3:227–244, 1993.
5. A. Datta and C. Icking. Competitive searching in a generalized street. In *Proceedings of ACM Symposium on Computational Geometry*, pages 175–182, 1994.
6. X. Deng, T. Kameda, and C. Papadimitriou. How to learn an unknown environment i: The rectilinear case. In *Proceedings of the 32nd IEEE Symposium on Foundation of Computer Science*, pages 298–303, 1991.
7. G. Dudek, K. Romanik, and S. Whitesides. Localizing a robot with minimum travel. In *Proceedings of the Sixth ACM-SIAM Symposium on Discrete Algorithms*, pages 437–446, 1995.
8. A. Ekman, A. Trone, and D. Stromberg. Exploration of polygonal environments using range data. *IEEE Transactions on Systems, Man, and Cybernetrics-Part B: Cybernetics*, 27(2):250–255, April 1997.
9. O. Faugeras. *Three-dimensional computer vision.* MIT Press, Cambridge, 1993.

10. G. Foux, M. Heymann, and A. Bruckstein. 2-dimensional robot navigation among unknown stationary polygonal obstacles. *IEEE Transactions on Robotics and Automation*, 9(1):96–102, February 1993.
11. S. K. Ghosh and J. W. Burdick. An on-line algorithm for exploring an unknown polygonal environment by a point robot. In *Proceedings of the Ninth Canadian Conference on Computational Geometry*, pages 100–105, 1997.
12. S. K. Ghosh and J. W. Burdick. Understanding discrete visibility and related approximation algorithms. In *Proceedings of the Ninth Canadian Conference on Computational Geometry*, pages 106–111, 1997.
13. S. K. Ghosh and J. W. Burdick. Exploring an unknown polygonal environment with a sensor based strategy. *Submitted for publication*, 2000.
14. S.K. Ghosh and S. Saluja. Optimal on-line algorithms for walking with minimum number of turns in unknown streets. *Computational Geometry: Theory and Applications*, 8:241–266, 1997.
15. C. Icking and R. Klein. Searching for the kernel of a polygon—a competitive strategy. In *Proc.* 11^{th} *Annu. ACM Symposium Comput. Geom.*, pages 258–266, Vancouver, Canada, June 1995.
16. I. Kamon, E. Rimon, and E. Rivlin. Tangentbug: A range sensor-based navigation algorithm. *International Journal of Robotics Research*, 17(9):934–953, September 1998.
17. I. Kamon and E. Rivlin. Sensory-based motion planning with global proofs. *IEEE Transactions on Robotics and Automation*, 13(6):814–822, December 1997.
18. S. H. Kim, J. H. Park, S. H. Choi, S. Y. Shin, and K. Y. Chwa. An optimal algorithm for finding the edge visibility polygon under limited visibility. *Information Processing Letters*, 53(6):359–365, March 1995.
19. R. Klein. Walking an unknown street with bounded detour. *Computational Geometry: Theory and Applications*, 1:325–351, 1992.
20. J. Kleinberg. On line search in a simple polygon. In *Proceedings of the fifth ACM-SIAM Symposium on Discrete Algorithms*, pages 8–15, 1994.
21. J. Leonard and H. F. Durrant-Whyte. *Directed sonar sensing for mobile robot navigation*. Kulwer Academic Publishers, Boston, MA, 1992.
22. J. O'Rourke. *Art Gallery Theorems and Algorithms*. Oxford University Press, New York, NY, 1987.
23. N.S.V. Rao. Robot navigation in unknown generalized polygonal terrains using vision sensors. *IEEE Transactions on Systems, Man, and Cybernetics*, 25(6):947–962, 1995.
24. T. Shermer. Recent results in art galleries. *Proc. of the IEEE*, 80(9):1384–1399, Sept. 1992.
25. I. Suzuki and M. Yamashita. Searching for a mobile intruder in a polygonal region. *SIAM Journal on Computing*, 21:863–888, 1992.
26. C. J. Taylor and D. J. Kriegman. Vision-based motion planning and exploration algorithms for mobile robots. *—EEE Transactions on Robotics and Automation*, 14(3):417–426, June 1998.
27. G. Fejes Tóth. Packing and covering. In *Handbook of Discrete and Computational Geometry, (edited by J. E. Goodman and J. O'Rourke)*, pages 19–42. CRC Press, 1997.

Parallel Optimal Weighted Links

Ovidiu Daescu

Department of Computer Science, University of Texas at Dallas,
Richardson, TX 75083, USA
E-mail: daescu@utdallas.edu.

Abstract. In this paper we consider parallel algorithms for computing an optimal link among weighted regions in the 2-dimensional (2-D) space. The weighted regions optimal link problem arises in several areas, such as geographic information systems (GIS), radiation therapy, geological exploration, environmental engineering and military applications. We present a CREW PRAM parallel algorithm and a coarse-grain parallel computer algorithm. Given a weighted subdivision with a total of n vertices, the *work* of the parallel algorithms we propose is only a $O(\log n)$ factor more than that of their (optimal) sequential counterparts.

1 The Weighted Regions Optimal Link Problem

We consider the (2-D) weighted regions optimal link problem, defined as follows: Given a subdivision R of the 2-D space, with m weighted regions R_i, $i = 1, 2, \ldots, m$, and a total of n vertices, find a *link* L such that: (1) L intersects two specified regions $R_s, R_t \in R$ and (2) the weighted sum $S(L) = \sum_{L \cap R_i \neq \phi} w_i * d_i(L)$ is minimized, where w_i is either the weight of R_i or zero and $d_i(L)$ is the length of L within region R_i. Depending on the application, the link L may be (a) unbounded (e.g., a line): the link L "passes through" the regions R_s and R_t; (b) bounded at one end (e.g., a ray): R_s is the source region of L and L passes through R_t and (c) bounded at both ends (e.g., a line segment): R_s is the source region of L and R_t is its destination region. We consider only straight links; the case of links described by some bounded degree curves is left for further study. Let R_L be the set of regions $\{R_{i_1}, \ldots, R_{i_k}\}$ intersected by a link L. Then, w_{i_1} and w_{i_k} are set to zero. This last condition assures that the optimal solution is bounded when a source (and/or a destination) region is not specified (cases (a) and (b)) and allows the link to originate and end arbitrarily within the source and target regions (cases (b) and (c)). See Figure 1 for an example.

The weighted regions optimal link problem is an extension of the optimal weighted penetration problem [5] and arises in several areas such as GIS, radiation therapy, stereotactic brain surgery, geological exploration, environmental engineering and military applications. For example, in military applications the weight w_i may represent the probability to be seen by the enemy when moving through R_i, from a secured source region R_s to another secured target region R_t. In radiation therapy, it has been pointed out that finding the optimal choice

V.N. Alexandrov et al. (Eds.): ICCS 2001, LNCS 2073, pp. 649–657, 2001.
© Springer-Verlag Berlin Heidelberg 2001

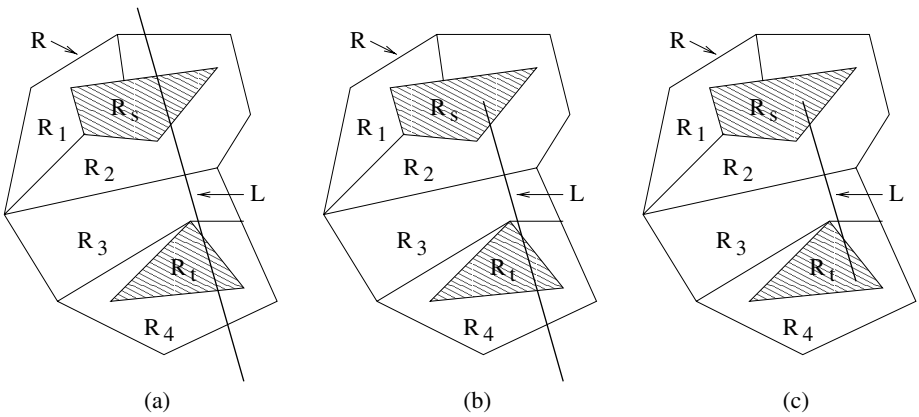


Fig. 1. Illustrating the problem: (a) L intersects R_s and R_t; (b) L originates in R_s and (c) L ends in R_t

for the link (cases (a) and (b)) is one of the most difficult problems of medical treatment optimization [3].

In computational geometry, there are a few results that consider weighted region problems, aiming to compute or approximate an optimal shortest path between pairs of points [1,15,16,17]. Mitchell and Papadimitriou [17] first considered the problem of computing an approximate geodesic shortest path between two points in a weighted planar subdivision. Their algorithm runs in $O(n^8 B)$ time and $O(n^4)$ space, where B is a factor representing the bit complexity of the problem instance, and approximates the optimal solution within an $(1+\epsilon)$ factor. Later, Mata and Mitchell [16] presented an approximation scheme for computing approximate shortest paths in a weighted polygonal subdivision. In $O(kn^3)$ time, they create a graph of size $O(kn)$ for $(1+\epsilon)$ approximate shortest paths, where k depends on ϵ and $0 \leq \epsilon < 1$: by varying the parameter k that controls the graph density, one can get arbitrarily close to the optimal solution.

The optimal link problem however, has a different structure than the shortest path problem. The few papers that consider finding an optimal link either discretize the problem (e.g., see [14]) or consider some simplified versions (e.g., see [18]). Important steps towards solving the optimal weighted penetration problem have been made very recently in [5,7], where it has been proved that the 2-D problem can be reduced to a number of (at most $O(n^2)$) subproblems, each of which asks to minimize a 2-variable function $f(x, y)$ over a convex domain D, where $f(x, y)$ is given as a sum of $O(n)$ terms. These subproblems can be generated sequentially in $O(n^2)$ time and thus the bulk of computation consists of solving the optimization problems. To compute the optimal solution for each subproblem, a global optimization software has been used in [5]. As the number of terms in f increases (i.e., > 100), such global optimization software performs badly in both time and memory usage. Since in practical applications, having

hundreds and even thousands of terms in the objective function is not an uncommon case, sequentially solving the global optimization problems (GOPs) on a single processor seems impractical. Instead, one can take advantage of the fact that, once the feasible domain and the objective function for each subproblem have been produced, the GOPs are independent and can be solved in parallel. After all GOPs are solved, the optimal solution can be obtained by a simple minimum selection.

It would then be of interests to efficiently produce the set of GOPs in parallel. We consider this problem and present the following results: (1) We give an $O(\log n)$ time, $O(n \log n + k)$ processors algorithm in the CREW PRAM model, where k is the total complexity description for the feasible domains of the GOPs ($\Omega(n^2)$ in the worst case). The algorithm is based on the arrangement sweeping techniques of Goodrich *et al.* [13]. Our parallel algorithm implies an optimal output sensitive $O(n \log n+k)$ time sequential algorithm for generating all GOPs, by using the optimal segment arrangement construction in [4]. (2) We show that, if at most n processors are available, all GOPs can be generated using $O(n^2 \log n)$ work. This algorithm is targeted to coarse-grain parallel computer models, consisting of a relatively small set of nodes (up to a few thousand), where each node has its own processor, with fair computing power, and a large local memory, allowing to store all data involved in (sequentially) solving the problem. In contrast, in a fine-grain computing model, one would allow only constant local memory, but unrestrict the number of processing nodes available.

2 Useful Structures

The optimal link problem can be reduced to solving a number of (at most $O(n^2)$) GOPs. Since each GOP can be solved using available global optimization software, we are only concerned with efficiently generating the GOPs. We start by describing the structure of a GOP.

Let L be a link intersecting the source and target regions R_s and R_t. Let S be the set of line segments in the subdivision R and let $S_{st} = \{s_{i_1}, s_{i_2}, \ldots, s_{i_k}\}$ be the subset of line segments in S that are intersected by L. Consider rotating and translating L. An event e_v will occur when L passes a vertex v of R. Such an event corresponds to some line segments (with an endpoint at v) entering or leaving S_{st}. As long as no event occurs, the formula describing the objective function $S(L)$ does not change and has the expression $S(L) = \sum_{i=i_1}^{i_{k-1}} w_i * d_i$, where d_i is the length of L inside region R_i and s_i, s_{i+1} are on the boundary of R_i. We refer the reader to [5,9] for more details.

Let $H = \{l_1, l_2, \ldots, l_n\}$ be a set of n straight lines in the plane. The lines in H partition the plane into a subdivision, called the *arrangement* $A(H)$ of H, that consists of a set of convex regions (cells), each bounded by some line segments on the lines in H. In general, $A(H)$ consists of $O(n^2)$ faces, edges and vertices and it can be computed in $O(n^2)$ time and $O(n)$ space, by sweeping the plane with a pseudoline [11].

For case (a) of the optimal link problem (the link L is a line), using a point-line duality transform that preserves the above/bellow relations (i.e., a point p above a line l dualizes to a line that is above the dual point of l), all lines intersecting the same subset of segments $S_{st} \in S$ correspond to a cell in the dual arrangement $A(R)$ of R, defined by $H_R = \{l_1, l_2, \ldots, l_n\}$, where $l_i \in H_R$ is the dual of vertex $v_i \in R$. The case of a semiline (case (b) of the link problem), and that of a line segment can be reduced to that of a line, by appropriately maintaining the set of line segments intersected by L and dropping those that arise before a segment in R_s or after a segment in R_t. This can be done sequentially in constant time, by extending the data structures in [5,9]. We leave the details to the full paper.

Generating and sweeping the entire arrangement however, as proposed in [5], may not be efficient since many cells of $A(R)$ may correspond to set of links that do not intersect R_s and/or R_t. Rather, we would like to compute only the cells of interest. Assume that R_s and R_t are convex (the results can be extended in the same complexity bounds to the nonconvex case, by observing that a line intersects a region R_i if and only if it intersects the convex hull of R_i; more details in the full version). Using a point-line duality transform that maps the line $y = mx + p$ in the (x,y) plane to the point (m, p) in the (m,p) plane, the set of lines intersecting R_s (resp., R_t), define a "strip" region D_{R_s} (resp. D_{R_t}) in between two m-monotone, unbounded and nonintersecting chains.

The set of lines intersecting both R_s and R_t thus correspond to the common intersection of D_{R_s} and D_{R_t}. Let k_s and k_t be the number of vertices of R_s and R_t, respectively. Let $D_{st} = D_{R_s} \cap D_{R_t}$.

Lemma 1. *D_{st} is a (possibly unbounded) region bounded by two m-monotone chains with a total of $O(k_s + k_t)$ vertices.*

Proof. D_{R_s} has k_s vertices, each vertex corresponding to a line supporting a boundary segment of R_s. Similarly, D_{R_t} has k_t vertices, each vertex corresponding to a line supporting a boundary segment of R_t. Since there are only $O(1)$ common tangents to R_s and R_t, the pairs of chains defining the boundaries of D_{R_s} and D_{R_t} intersect $O(1)$ times, and the proof follows. □

An example is given in Figure 2, where D_{st} is the quadrilateral with vertices A,B,C and D.

Lemma 2. *The lines in $A(R)$ have at most $O(n)$ intersections with the chains bounding D_{st}.*

Proof. Only $O(1)$ lines tangent to R_s and R_t can pass through a point p. Then, the dual line of p can intersect the chains bounding D_{st} only $O(1)$ times, from which the proof follows. □

Thus, computing the cells of the arrangement defined by $A(R)$ that correspond to set of lines intersecting both R_s and R_t reduces to computing the arrangement of $O(n)$ line segments in D_{st} (some of these line segments may in fact be semilines, but this does not influence the overall computation).

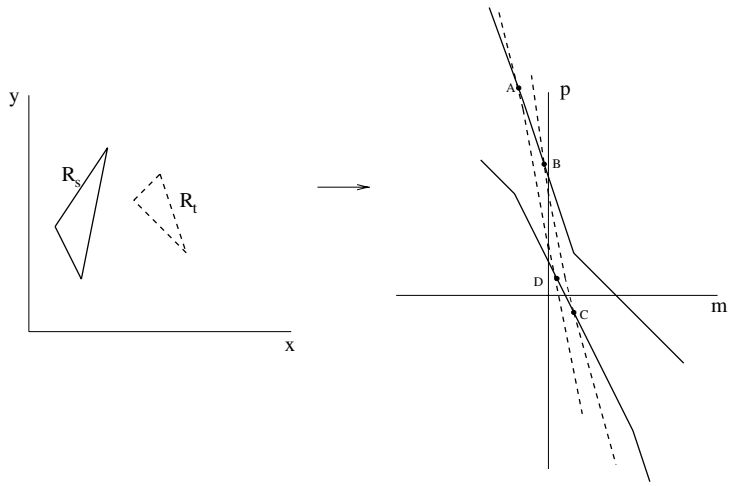


Fig. 2. The line transversals of R_s, R_t dualize to quadrilateral D_{st}=ABCD

3 Parallel Solutions

In this section we present two parallel solutions for the optimal link problem. The first algorithm uses the CREW PRAM model of computation. Recall that in this model processors act synchronously and may simultaneously access for reading the same memory location on a shared memory space. To obtain output sensitive algorithms, we use the paradigm in [13]: the pool of virtual processors can grow as the computation proceeds, provided that the allocation occurs globally [12]. Given a subdivision R with a total of n vertices, the algorithm we present runs in $O(\log n)$ time using $O(n \log n + k)$ processors, where k is the size of the output (the total description complexity for the feasible domains of the GOPs to be solved), and it could be $\Omega(n^2)$ in the worst case. If the traditional CREW PRAM model is used, our solution would require $O(n^2)$ processors.

As outlined in the previous section, to compute the feasible domains for the GOPs it suffices to compute the cells in the arrangement $A(D_{st})$ of $O(n)$ line segments in D_{st}, where each line segment has its endpoints on the boundary of D_{st}. Further, in order to produce the corresponding objective functions, with each cell C of $A(D_{st})$ we must associate the subset of line segments in S that are intersected by a line whose dual is a point in C. This computation may be regarded as a set of queries on the line segments in S.

The algorithm we present follows the one in [13], where the following segment intersection problem has been considered and solved: given a set of line segments in the plane, construct a data structure that allows to quickly report the segments intersected by a query line. Their algorithm is based on a parallel persistence data structure termed *array-of-trees* and on fast construction of line arrangements. The main idea in [13] is to build the arrangement, an operation sequence σ for

that arrangement, and then use the array-of-trees data structure to evaluate the sequence. A reporting query can then be answered in $O(\log n)$ time per query, resulting in a $O(\log n)$ time, $O(n^2)$ processors CREW PRAM algorithm.

The main difference in the algorithm we present is in defining and handling the operation sequence σ. Given the nature of the optimal link problem, a vertex of the subdivision R may in fact be the endpoint of multiple line segments (e.g., $O(n)$ such segments). Then, while crossing from one cell to an adjacent one, many line segments may enter or leave the set S_{st} and thus many *enable/disable*-like operations in [13] would be associated to such crossing. Rather than defining the enable/disable operations on individual segments, we define these operations on subsets of segments in S. Doing this, in order to maintain the processing bounds, we must be able to obtain these subsets in constant time per subset. Fortunately, this can be done by extending the data structures introduced in [9, 5] for the optimal penetration problem. We only mention here that, if not given as part of the input, the additional data structures can be easily computed in parallel in $O(\log n)$ time using $O(n)$ processors. Knowing the number $d(v)$ of edges adjacent to each vertex $v \in R$ and using these structures, we can assign $O(d(v))$ processors to handle an event at v in constant time. Observe that, since R is a planar subdivision, we have $\sum_{v \in R} d(v) = O(n)$.

Lemma 3. *The feasible domains and the objective functions for the GOPs associated with the region D_{st} can be generated in $O(\log n)$ time using $O(n \log n + k)$ processors, where k is the size of the output.*

Proof. We give an algorithm that constructs the GOPs in the claimed time and processor bounds. The algorithm proceeds as follows. (1) Construct the arrangement of line segments inside D_{st}. This can be done in $O(\log n)$ time with $O(n \log n + k)$ processors, using the algorithm in [12]. We then compute a spanning tree for this arrangement and an Euler tour of this tree, as in [13]. While computing the Euler tour, we use an extension of the data structures in [9, 5] to produce the operation sequence σ for the tour. Since the enable/disable operations in σ add only constant time, this computation can still be done in $O(\log n)$ time using $O(k/\log n)$ processors. Constructing the array-of-trees data structure and answering reporting queries can be done as in [13]. Then, the claimed processing bounds follow. □

We mention here that an $O(\log n)$ time, $O(n^2)$ processors algorithm can be obtained by associating an enable/disable operation with each line segment involved in a crossing at a node v (i.e., to $O(d(v))$ segments) and applying the algorithm in [13].

The second algorithm we present uses a coarse-grain parallel computer model of computation. In this model, a relatively small number of processors are available and each processor has a large amount of local memory available, thus being able to store all data involved in (sequentially) solving the problem, much like a personal computer. In particular, such a processing element would be able to store the region R and its dual arrangement, as well as all data that is required in the process of generating and solving a GOP. If at most n processors are available, we present a simple yet efficient algorithm that generates all GOPs using

$O(n^2 \log n)$ work and with practically no communications between processors. The GOPs can be solved locally or they can be sent for solving to some external processing clusters, as in [10].

We make the following assumptions for our model: (1) processors are connected and can communicate via a global data buss or a communication network that allows efficient data broadcasting (i.e, feed the subdivision R to all processing elements) and (2) processors are numbered and each processor knows its order number.

The algorithm we present is based on computing the portion of an arrangement of lines that lies in between two vertical lines. At the start of the algorithm, each processing element stores the subdivision R and the set of lines in $A(R)$ (following a broadcasting operation), and knows its order number. Since each processor will perform similar computation, it then suffices to discuss the computation involved at only one of them, say the k-th processor P_k.

At processor P_k, the algorithm will compute the GOPs associated with the portion of the arrangement $A(R)$ that is in between the vertical lines L_{k-1} and L_k passing through the $(k-1)n$-th and kn-th leftmost intersection points of the lines in $A(R)$. We denote these two points as p_{k-1} and p_k. First, the algorithm finds the lines L_{k-1} and L_k by computing the points p_{k-1} and p_k. These points can be computed in $O(n \log n)$ time each using the algorithm in [8]. Next, the algorithm computes the intersection points of the lines in $A(R)$ with L_{k-1} and L_k and runs a topological sweep algorithm [2] to produce the GOPs inside the parallel strip. Sweeping the strip, as well as generating the corresponding objective functions, can be done altogether in $O(n \log n)$ time, which follows from [9,5]. Alternatively, we can obtain the same results using the (optimal) sequential version of the CREW PRAM algorithm above (i.e., by computing a line segment arrangement inside the strip and traversing that arrangement). Finally, the last step of the algorithm consists of a maximum selection among the optimal solutions stored "locally" at different processing elements, in order to obtain the optimum over all GOPs. These can be done using $O(n)$ broadcasting operations, starting at processor P_1, with the overall optimum computed at processor P_n. Thus, we have the following lemma.

Lemma 4. *In the proposed coarse-grain computing model, the feasible domains and the objective functions for the GOPs can be computed in $O(n \log n)$ time using $O(n)$ processors.*

Corollary 1. *If only p processors are available, where $p \leq n$, the feasible domains and the objective functions for the GOPs can be computed with $O(n^2 \log n)$ total work.*

There are two important features of our solution that should be noted here. First, the approach we propose allows for scalability in solving the GOPs. That is, after a GOP is produced, it can be solved either locally or it can be sent to some external processing cluster, that would in turn compute and return the optimal value for that GOP. Second, once the initial setup for the computation has been completed, it takes constant time to generate a new GOP; since the

objective function of a GOP could have $O(n)$ terms, this implies that all GOPs in a strip can be generated in time comparable to that required to perform a single evaluation of a GOP's objective function, and justifies the proposed coarse-grain model of computation.

In the full paper, we will show that the algorithm above can be extended to compute only the GOPs corresponding to the portion of the arrangement $A(R)$ that lies inside the region D_{st}, with each processing element solving about the same number of GOPs. However, we expect such an approach to be slower in practice when compared to the algorithm above, due to the increased complexities of the data structures involved, which may considerably add to the values of the constants hidden in the big-Oh notations.

References

1. L. Aleksandrov, M. Lanthier, A. Maheshwari, and J.-R. Sack, "An ϵ-approximation algorithm for weighted shortest paths on polyhedral surfaces," *Proc. of the 6th Scandinavian Workshop on Algorithm Theory*, pp. 11-22, 1998.
2. T. Asano, L.J. Guibas and T. Tokuyama, "Walking in an arrangement topologically," *Int. Journal of Computational Geometry and Applications*, Vol. 4, pp. 123-151, 1994.
3. A. Brahme, "Optimization of radiation therapy," *Int. Jouurnal of Radiat. Oncol. Biol. Phys.*, Vol. 28, pp. 785-787, 1994.
4. B. Chazelle and H. Edelsbrunner, "An optimal algorithm for intersecting line segments in the plane," *Journal of ACM*, Vol. 39, pp. 1-54, 1992.
5. D.Z. Chen, O. Daescu, X. Hu, X. Wu and J. Xu, "Determining an optimal penetration among weighted regions in two and three dimensions," *Proceedings of the 15th ACM Symposium on Computational Geometry*, pp. 322-331, 1999.
6. D.Z. Chen, O. Daescu, Y. Dai, N. Katoh, X. Wu and J. Xu, "Optimizing the sum of linear fractional functions and applications," *Proceedings of the 11th ACM-SIAM Symposium on Discrete Algorithms*, pp. 707-716, 2000.
7. D.Z. Chen, X. Hu and J. Xu, "Optimal Beam Penetration in Two and Three Dimensions," *Proceedings of the 11th Annual International Symposium on Algorithms And Computation*, pp. 491-502, 2000.
8. R. Cole, J. Salowe, W. Steiger and E. Szemeredi, "Optimal Slope Selection," *SIAm Journal of Computing*, Vol. 18, pp. 792-810, 1989.
9. O. Daescu, "On Geometric Optimization Problems", PhD Thesis, May 2000.
10. O. Daescu, "Optimal Link Problem on PIMs", Manuscript, January 2001.
11. H. Edelsbrunner, and L.J. Guibas, "Topologically sweeping an arrangement," *Journal of Computer and System Sciences* Vol. 38, pp. 165-194, 1989.
12. M Goodrich, "Intersecting Line Segments in Parallel with an Output-Sensitive Number of Processors," *SIAM Journal on Computing*, Vol. 20, pp. 737-755, 1991.
13. M Goodrich, M.R. Ghouse and J. Bright, "Sweep methods for Parallel Computational Geometry," *Algorithmica*, Vol. 15, pp. 126-153, 1996.
14. A. Gustafsson, B.K. Lind and A. Brahme, "A generalized pencil beam algorithm for optimization of radiation therapy," *Med. Phys.*, Vol. 21, pp. 343-356, 1994.
15. M. Lanthier, A. Maheshwari, and J.-R. Sack, "Approximating weighted shortest paths on polyhedral surfaces," *Proc. of the 13th ACM Symp. on Comp. Geometry*, pp. 274-283, 1997.

16. C. Mata, and J.S.B. Mitchell, "A new algorithm for computing shortest paths in weighted planar subdivisions," *Proc. of the 13th ACM Symp. on Comp. Geometry*, pp. 264-273, 1997.
17. J.S.B. Mitchell and C.H. Papadimitriou, "The weighted region problem: Finding shortest paths through a weighted planar subdivision," *Journal of the ACM*, Vol. 38, pp. 18-73, 1991.
18. A. Schweikard, J.R. Adler and J.C. Latombe, "Motion planning in stereotaxic radiosurgery," *IEEE Trans. on Robotics and Automation*, Vol. 9, pp. 764-774, 1993.
19. J. Snoeyink and J. Hershberger, "Sweeping Arrangements of Curves," *DIMACS Series in Discrete Mathematics*, Vol. 6, pp. 309-349, 1991.

Robustness Issues in Surface Reconstruction

Tamal K. Dey, Joachim Giesen, and Wulue Zhao*

Abstract. The piecewise linear reconstruction of a surface from a sample is a well studied problem in computer graphics and computational geometry. A popular class of reconstruction algorithms filter a subset of triangles of the three dimensional Delaunay triangulation of the sample and subsequently extract a manifold from the filtered triangles. Here we report on robustness issues that turned out to be crucial in implementations.

1 Introduction

While implementing geometric algorithms, one often has to face the problem of numerical instabilities. That is also the case for Delaunay based surface reconstruction algorithms that filter a subset of Delaunay triangles for reconstruction. But careful examination shows that the only step that inherently requires instable numerical decisions is the construction of the Delaunay triangulation itself. All other steps can be implemented relying either on numerically stable or purely combinatorial decisions. Here we want to emphasize the following design principle for geometric implementations: *Avoid numerical decisions whenever possible.* Our experience shows this pays off as well in robustness as in running time.

2 Filter Based Reconstruction Algorithms

Filter based algorithms consider a subset of triangles of the three dimensional Delaunay triangulation of a sample $P \subset \mathbb{R}^3$ for reconstruction. All these algorithms contain three generic steps:

(1) FilterTriangles. A set of *candidate* triangles is extracted from the Delaunay triangulation of the the sample. In general the underlying space of these triangles is not a manifold, but a manifold with boundary can be extracted.
(2) Pruning. We want to extract a manifold from the set of candidate triangles by walking either on the inside or outside of this set. During the walk we may encounter the problem of entering a triangle with a bare edge, i.e. an edge with only one incident triangle. The purpose of this step is to get rid of such triangles.
(3) Walk. We walk on the in- or outside of the set of triangles that remained after Pruning and report the triangles walked over.

Different filter based reconstruction algorithms distinguish themselves in the FilterTriangles step. In the following we shortly explain two different filter strategies which both come with theoretical guarantees. But there are also other algorithms that fit in the general scheme presented above.

* Department of CIS, Ohio State University, Columbus, OH 43210. This work is supported by NSF grant CCR-9988216.

V.N. Alexandrov et al. (Eds.): ICCS 2001, LNCS 2073, pp. 658–662, 2001.
© Springer-Verlag Berlin Heidelberg 2001

2.1 Crust

The CRUST algorithm of [1] first computes the Voronoi diagram of the sample P, i.e. the dual of the Delaunay triangulation. A subset of the Voronoi vertices called *poles* is used to filter Delaunay triangles.

Poles: Let V_p be the Voronoi cell of a sample point $p \in P$. The Voronoi vertex p^+ in the Voronoi cell V_p farthest from p is called the positive pole of p. The negative pole of p is the point $p^- \in V_p$ farthest from p such that the two vectors $(p^+ - p)$ and $(p^- - p)$ make an angle more than $\frac{\pi}{2}$. We call $\mathbf{v_p} = p^+ - p$ the pole vector of the sample p. See Figure 1. If V_p is unbounded special care has to be taken.

The CRUST algorithm computes the Delaunay triangulation of the union of the sample P with the set of poles. All triangles in this Delaunay triangulation that are incident to three samples from the original sample P are candidate triangles for the reconstruction.

2.2 Cocone

The COCONE algorithm of [2,4] avoids the second Delaunay computation. This algorithm is using a set called *cocone* for every sample point $p \in P$ to filter Delaunay triangles.

Cocone: The set $C_p(\theta) = \{y \in V_p \ : \ \angle((y-p), \mathbf{v_p}) \geq \frac{\pi}{2} - \theta\}$ is called the cocone of p. In words, $C_p(\theta)$ is the complement of a double cone centered at p (clipped within V_p) with opening angle $\frac{\pi}{2} - \theta$ around the axis aligned with $\mathbf{v_p}$. See Figure 1.

The COCONE algorithm filters a triangle t from the Delaunay triangulation of the sample P if all cocones of the three sample points incident to t intersect the Voronoi edge dual to t.

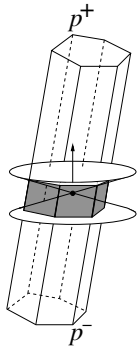


Fig. 1. A Voronoi cell together with the normalized pole vector and the cocone.

3 Robustness

In this section we discuss the robustness of the four steps (including the computation of the Delaunay triangulation / Voronoi diagram) of the generic filter based algorithm.

3.1 Delaunay Triangulation

Delaunay triangulation algorithms are usually designed for the real RAM, a random access machine that can handle real numbers at unit cost. Most of these algorithms assume that two geometric predicates, the *sidedness* test and the *incircle* test, can be evaluated accurately. The sidedness test decides whether a point lies left of, right of or on an oriented hyperplane. The incircle test decides whether a point lies outside of, inside of or on a sphere. Both predicates amount to the computation of the sign of a determinant. Implementing these tests using floating point arithmetic can result in completely unreliable output or even infinite loops depending on the chosen algorithm.

The naive way to circumvent these problems is to compute the value of the determinants using exact arithmetic and to read of the sign from the value. A more efficient technique is the use of floating point filters. A floating point filter computes an approximate value of an expression and a bound for the maximal deviation from the true value. If the error bound is smaller than the absolute value of the approximation, approximation and exact value have the same sign. In this case we can use the sign of the approximation to decide the predicate.

In our implementations we used the floating point filters provided by the computational geometry algorithms library CGAL [3]. Our experience shows that the running time is no more than twice the running time of a pure floating point implementation. See Figure 2(a) for an example how the use of floating point arithmetic can affect the reconstruction algorithms (after FilterTriangles).

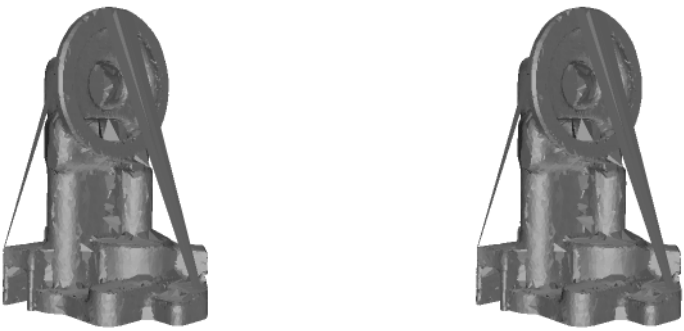


Fig. 2. Candidate Triangles computed by the Cocone algorithm from a Delaunay triangulation computed with floating point arithmetic (left) and filtered exact arithmetic (right).

3.2 Filter Triangles and Pruning

The step FilterTriangles is purely combinatorial in the Crust algorithm and hence robust. In the Cocone algorithm this step involves the numerical decision if a Voronoi edge intersects a cocone. But it turns out that the exact size of the opening angle of the cocone is not important. Thus the decision if a Voronoi edge intersects a cocone need not be really accurate.

The PRUNING step is purely combinatorial. It involves only the decision if an edge is bare, i.e. if it has exactly one incident triangle. Hence this step is also robust.

3.3 Walk

A pseudo code for the implementation of the walk is given below.

```
WALK (C, (t, e))
 1  S := {t}
 2  Pending := ∅
 3  push (t, e) on Pending.
 4  while Pending ≠∅
 5    pop (t, e) from Pending
 6    if e is not marked processed
 7      mark e processed.
 8      t' := SURFACENEIGHBOR (C, t, e)
 9      S := S ∪ {t'}
10      if e' ≠−e incident to t' induces the same orientation on t' as −e
11        push (t', e') on Pending.
12  return S
```

The WALK takes two parameters, a complex C containing the candidate triangles and an oriented triangle t. The orientation of t is given by an oriented edge $\boldsymbol{e}$ incident to t. First, the surface S is initialized with the triangle t (line 1). Next a stack *Pending* is initialized with the oriented triangle t (lines 2 and 3). As long as the stack *Pending* is not empty, we pop its top element $(t, \boldsymbol{e})$. If the edge $\boldsymbol{e}$ is not already processed we call the function SURFACENEIGHBOR to compute the surface neighbor of the oriented triangle t, i.e. the triangle t' that 'best fits' t (line 8). Then t' is inserted in S and two new oriented triangles are pushed on the stack *pending* (lines 9 to 11). Finally we return S (line 12).

The question is how to implement the function SURFACENEIGHBOR which has to circle around edge $\boldsymbol{e}$ according to the orientation of $\boldsymbol{e}$ until it first encounters another candidate triangle. This is the triangle we are looking for. Let t' always denote a candidate triangle incident to t via $\boldsymbol{e}$.

A naive implementation could compute first the value $(\boldsymbol{n}_{t'} \times \boldsymbol{n}_t) \cdot \boldsymbol{e}$ for every triangle t'. Here $\boldsymbol{n}_t$ and $\boldsymbol{n}_{t'}$ denote the normalized normals of t and t' both oriented according to the orientation of t. From the sign of this value one can decide if t' and t lie on the same side of the hyperplane h_1 spanned by the vectors $\boldsymbol{n}_t$ and $\boldsymbol{e}$. Next the value $\lambda_{t'} = (\boldsymbol{v}_{t'} \cdot \boldsymbol{n}_t)$ is computed. Here $v_{t'}$ denotes the normalized vector from the head of $\boldsymbol{e}$ to the vertex opposite of $\boldsymbol{e}$ in t'. See Figure 3(a). Using the sign of $\lambda_{t'}$ one can decide if t' lies above or below the oriented hyperplane h_2 defined by t. In case that there exists a triangle t' which lies above h_2 and on the same side of h_1 as t the function SURFACENEIGHBOR returns the triangle which has the smallest value $\lambda_{t'}$ among all such triangles. Otherwise it returns the triangle which has the largest value $\lambda_{t'}$ among all triangles t' that do not lie on the same side of h_1 as t. If such a triangle does not exist

the function just returns the triangle t' which has the smallest value $\lambda_{t'}$. The WALK with this implementation of SURFACENEIGHBOR can produce holes due to numerical inaccuracy in the computation of $\lambda_{t'}$ when walking over *slivers*, i.e. flat tetrahedra which frequently appear in the Delaunay triangulation of surface samples. See Figure 3(b).

Fig. 3. An instable way to compute the surface neighbor (left) and a zoom on a reconstruction after the WALK (right).

A robust and faster implementation of the function SURFACENEIGHBOR avoids numerical computations by exploiting the combinatorial structure of the Delaunay triangulation to choose the next triangle. Every triangle in the Delaunay triangulation has two incident tetrahedra. We fix a global orientation. For the triangle t we choose the tetrahedron that is oriented according to the orientation of $(t, \boldsymbol{e})$ and the global orientation. In Figure 3(b) this is tetrahedron T_1. Then we go to neighboring tetrahedra $T_2, T_3, \ldots$ also incident to $\boldsymbol{e}$ until we find the triangle t'. See Figure 4(a). The WALK with this implementation of SURFACENEIGHBOR is robust since no numerical decisions are involved. The latter is also the reason why it is fast provided the Delaunay triangulation is given in a form which allows to answer queries for neighboring tetrahedra quickly. With our implementation we observe that the time spend for the WALK is only a tiny fraction of the time needed to compute the Delaunay triangulation.

Fig. 4. A stable way to compute the surface neighbor (left) and a zoom on a reconstruction after the WALK (right).

References

1. N. Amenta and M. Bern. Surface reconstruction by Voronoi ltering. *Discr. Comput. Geom.*, **22**, (1999), 481 504.
2. N. Amenta, S. Choi, T. K. Dey and N. Leekha. A simple algorithm for homeomorphic surface reconstruction. *Proc. 16th. ACM Sympos. Comput. Geom.*, (2000), 213 222.
3. `http://www.cgal.org`
4. T. K. Dey and J. Giesen. Detecting undersampling in surface reconstruction. *Proc. 17th ACM Sympos. Comput. Geom.*, (2001), to appear.

On a Nearest-Neighbour Problem in Minkowski and Power Metrics

M.L. Gavrilova

Dept of Comp. Science, University of Calgary
Calgary, AB, Canada, T2N1N4
marina@cpsc.ucalgary.ca

Abstract. The paper presents an efficient algorithm for solving the nearest-neighbor problem in the plane, based on generalized Voronoi diagram construction. The input for the problem is the set of circular sites S with varying radii, the query point p and the metric (Minkowski or power) according to which the site neighboring the query point, is to be reported. The IDG/NNM software was developed for an experimental study of the problem. The experimental results demonstrate that the Voronoi diagram method outperforms the $k-d$ tree method for all tested input site configurations. The similarity between the nearest-neighbor relationship in the Minkowski and power metrics was also established.

1 Introduction

The Voronoi diagram is often used as a convenient tool for solving scientific problems in computer modeling of physical phenomena. These include structure analysis of unordered systems (liquids, solutions, polymers) [12], stress analysis and simulation of granular systems (ice flow, silo models) [9], and space structures in complex molecular and biological systems [11,17].
There are some challenges arising while investigating such problems. Existing software and algorithms are not customized to efficiently solve a variety of application problems. A particular problem addressed in this paper is the finding of a nearest-neighbor in a system of poly-sized circular objects [14]. The application of the ordinary point site Voronoi diagram to perform the nearest-neighbor query in 2D is straightforward [1]. The algorithm takes $O(n)$ space, $O(nlogn)$ preprocessing time, and the worst-case running time of $O(n)$. The same idea has been extended to higher dimensions [3], applied to solve the point location problem among the convex sites [16,10], and used to solve the nearest-neighbor problem for dynamic Voronoi diagram [4]. The generalized VD in Laguerre geometry was successfully used to solve the collision optimization problem in a system of moving particles [9]. The properties of the generalized weighted Voronoi diagrams that enable the use of this data structure for nearest-neighbor detection were thoroughly investigated in [8]. However, there has not been a study that compares the various generalized Voronoi diagrams with respect to solving the nearest-neighbor problem.
Thus, this paper presents a study of the generalized Voronoi diagram approach

V.N. Alexandrov et al. (Eds.): ICCS 2001, LNCS 2073, pp. 663–672, 2001.
© Springer-Verlag Berlin Heidelberg 2001

for finding the nearest-neighbor for a set of non-intersecting circles. Application domains for this problem can be found in computer graphics, GIS, computer modeling and computer simulation [14]. The developed method can also be applied to problems from statistics and information retrieval fields. The data structures studied are the generalized weighted Voronoi diagram (VD) under the Manhattan, supremum and power metrics. The main result is an efficient and robust algorithm for the nearest-neighbor computation in Manhattan, supremum and power metrics.
The performance of the VD based method was compared against the CORE library implementation of the $k-d$ tree method [2] that was modified to handle weighted sites under the supremum metric. The experimental results show significantly better performance for the generalized VD based method, including large (10,000 sites) data sets with various topologies. It is also worth noting that a similar technique can be applied to solve a variety of problems, such as all nearest-neighbors, point location and range queries.

2 Problem Definition

Consider a set of circular sites S in the plane. Define a *nearest-neighbor* relation between the query point x and a site P as follows. The point $x \in R^2$ is the nearest neighbor of $P \in S$ iff $d(x,P) \leq \min_{Q \neq P} d(x,Q)$, $Q \in S$. The distance $d(x,P)$ between a point $x(x_1,x_2)$ and a circle $P = \{p, r_p\}$ with the center at $p(p_1,p_2)$ and radius r_p can be computed as

$$d(x,P) = d(x,p) - r_p = |x_1 - p_1| + |x_2 - p_2| - r_p \tag{1}$$

in the Manhattan (L_1) metric, and as

$$d\,(x,P) = d(x,p) - r_p = \max\left(|x_1 - p_1|\,, |x_2 - p_2|\right) - r_p \tag{2}$$

in the supremum (L_∞) metric. In Laguerre geometry (under the power distance function) it is computed according to the formula:

$$d(x,P) = d(x,p)^2 - r_p^2 = (x_1 - p_1)^2 + (x_2 - p_2)^2 - r_p^2. \tag{3}$$

We propose to use the generalized Voronoi diagram as a tool to solve the problem. The *generalized Voronoi diagram* of a set of circles S in the plane is a set of generalized Voronoi regions $GVor(P)$, where

$$GVor\,(P) = \{\, x|\ \ d\,(x,P) \leq d\,(x,Q)\,, \forall Q \in S - \{P\}\} \tag{4}$$

and $d(x,P)$ is the distance function between a point x and a circle P [14]. The distance function $d(x,P)$ is defined according to the metric employed. The example of the generalized Voronoi diagram in supremum metric of 1000 sites can be found in Fig. 1. A *generalized Delaunay triangulation* (DT) is the dual of a generalized Voronoi diagram obtained by joining all pairs of sites whose generalized Voronoi regions share a common edge. The nearest-neighbor property

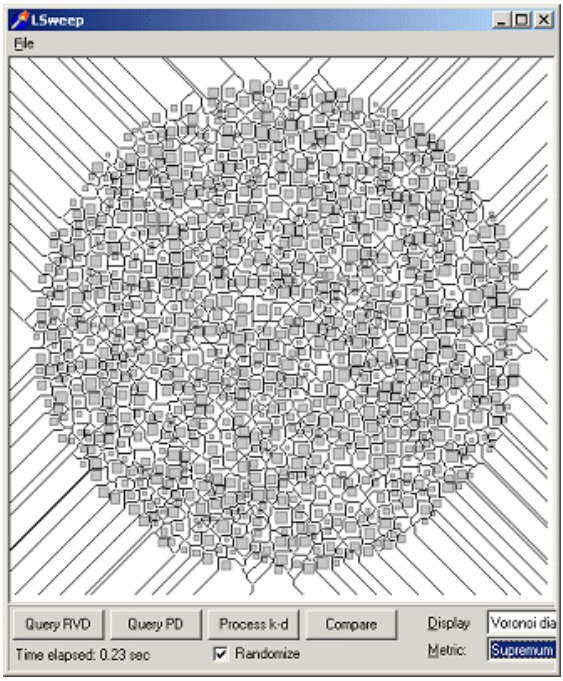


Fig. 1. Supremum weighted VD for 1000 randomly distributed sites generated using the Initial Distribution Generator (IDG) module.

for VD in various metrics, including Manhattan, supremum and power, was established in [8]. This property allows solving the nearest-neighbor problem by constructing the generalized Voronoi diagram or the Delaunay triangulation. The method is presented in the following section.

3 The Nearest-Neighbor Search Algorithm

The following outlines the nearest-neighbor search algorithm based on the generalized DT construction. The input for the algorithm is a set of circular sites. The approach is based on the simple edge-walk technique that starts with a random location in the Delaunay triangulation:

1. (Initialization) Build the weighted generalized Delaunay triangulation (using, for example, a flip-based incremental construction technique).
2. Find a site P neighboring the query point x.
   a) Randomly select a site P_0 as a starting site for the search (call it the *active* site).

b) Randomly select an edge adjacent to the active site in the Delaunay triangulation and set its value to $e_{prev} = e_{curr} = e_{new}$

c) Perform a counter-clock wise walk alone the DT edges adjacent to the active site. Select the first edge such that x is located to the left of the straight line passing through this edge, by performing the CCW orientation test.

d) Update $e_{prev} = e_{curr}$, $e_{curr} = e_{new}$. Set e_{new} as the newly found edge.

e) If edges e_{prev},e_{curr}, e_{new} do not form a triangle enclosing the query point x, set the endpoint of e_{new} edge to be the new active site. GOTO 2(c).

3. Report the closest of the vertices of the triangle formed by the edges e_{prev},e_{curr}, e_{new} as the nearest-neighbor to the query point x.

The preprocessing step is the worst-case optimal $O(nlogn)$, the worst-case number of edges visited during the Step 2 is $O(n)$ (since we never consider a visited edge twice) and the space complexity is $O(n)$.
Note 1: The incremental flipping algorithm description can be found in [5].
Note 2: The algorithm is applicable for solving the point location problem and the range search problem in the presence of specific constraints. Thus, the presented algorithm locates the Voronoi region containing the query point and the generator of this Voronoi region is reported as the nearest neighbor.
Note 3: The Voronoi diagram does not depends on the sizes or distributions of the circles, with the exception of close to degenerate cases, that also require special treatment in the cell-based or $k-d$ tree methods [2].

4 IDG/NNM Software

The algorithm outlined above was implemented in the object-oriented Borland Delphi environment. The experiments were conducted on a Pentium II 350 computer with 128 MB RAM. The program consists of two modules. The first module, the Initial Distribution Generator (IDG), is used to create various configurations for the input sites. IDG can generate a new distribution by importing the distribution from a text file, where the coordinates of the centers and radii of circles are specified. IDG can also automatically generate various distributions, such as uniform distribution of sites in the interior of a square, uniform distribution of sites in the interior of a circle, cross, ring, degenerate grid and degenerate circle (see Fig. 2). The parameters of the distribution, including the number of circles, the distribution of their radii, the size of the area, and the type of the distribution must be specified as well. The second module, the Nearest-Neighbor Monitor (NNM), is the program that constructs the additively weighted supremum VD, the power diagram and the $k-d$ tree in supremum metric for the

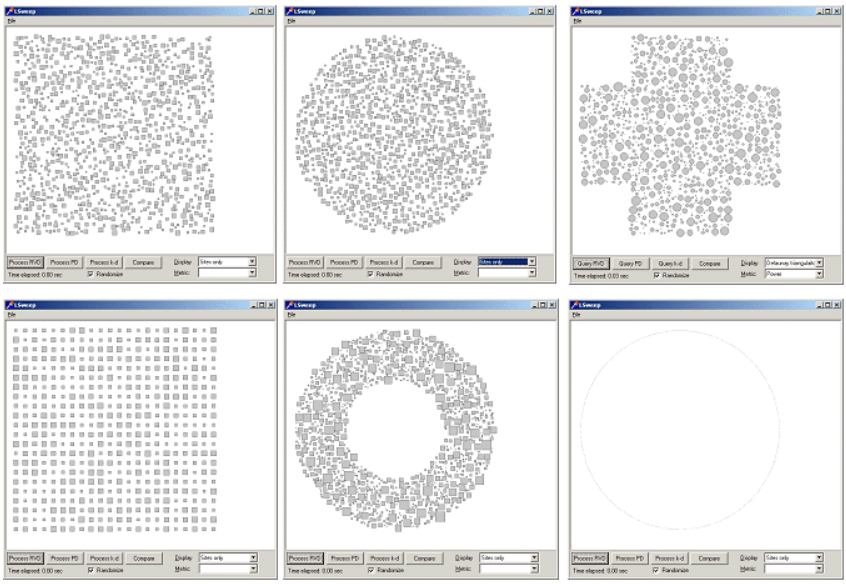


Fig. 2. Six configurations of sites in supremum metric (left right, top down direction): uniform square, uniform circle, cross, degenerate grid, ring and degenerate circle.

specified input configuration. Then NNM performs a series of nearest-neighbor searches.

The efficiency of the VD-based method was compared against the $k-d$ tree method for a set of circles in the plane. The $k-d$ tree implementation is based on the Ranger software [13], which implements an optimal $k-d$ tree method for a set of points [6]. The software was modified to accommodate the circular sites. Each site was represented by the four corner points, effectively allowing reporting the nearest neighbor in the weighted supremum metric. The software was also optimized to avoid the unnecessary memory allocations and initializations of the internal variables for maximum efficiency when performing multiple queries. The efficiency of the method does not depend on the metric used, thus the supremum and power VD methods were compared to the same implementation of the $k-d$ tree method.

After the initial distribution is generated, it is loaded into the NNM Module. First, the generalized Voronoi diagram, Delaunay Triangulation or a $k-d$ tree is computed in power or supremum metric. The snapshot of the screen (see Fig. 3) illustrates the Voronoi diagram in Laguerre geometry of 10000 circular sites in ring configuration. The nearest-neighbor queries are done by either generating a sequence of random queries or by selecting the query point manually. When a manual query is performed, the path from the starting VD edge to the nearest-neighbor of the query point is highlighted and the length of this path is displayed.

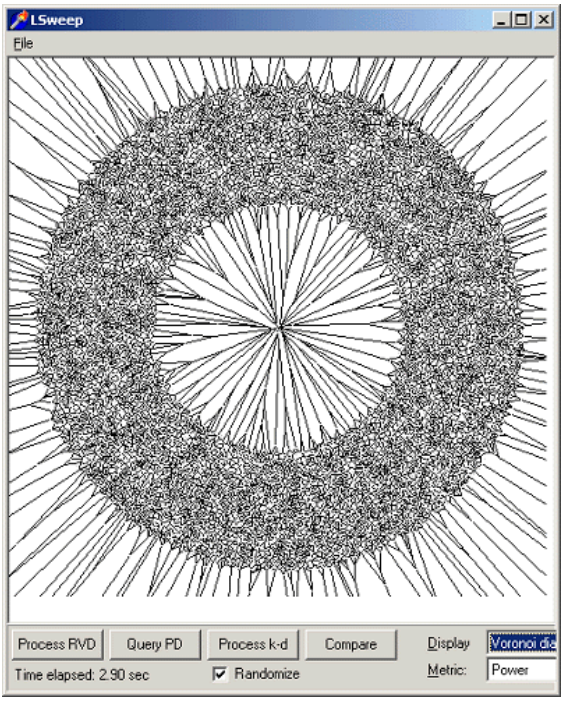


Fig. 3. Example of generalized VD under power metric for 10000 sites, ring configuration

The measured characteristics that represent the performance of the method include the total number of queries performed, the elapsed time, the average time per query, and the *average search length*. The average search length is a parameter related to the number of comparisons that the algorithm performs. For the VD approach, this parameter represents the total number of edges that were encountered during the edge walk while performing a query. In case of $k-d$ tree this parameter represents the total number of distance comparisons performed on different nodes of the tree. This parameter was selected for evaluation since it helps to compare VD and $k-d$ tree methods, and it can be easily visualized.

5 Experimental Studies

The experiments were performed for different data set sizes, various distributions of their density, radii and site configurations. The algorithms were tested on the generated data sets and the data set representing granular-type material system for a silos simulation with large number of particles (the data sets were provided by the Department of Mechanical Engineering, University of Calgary).

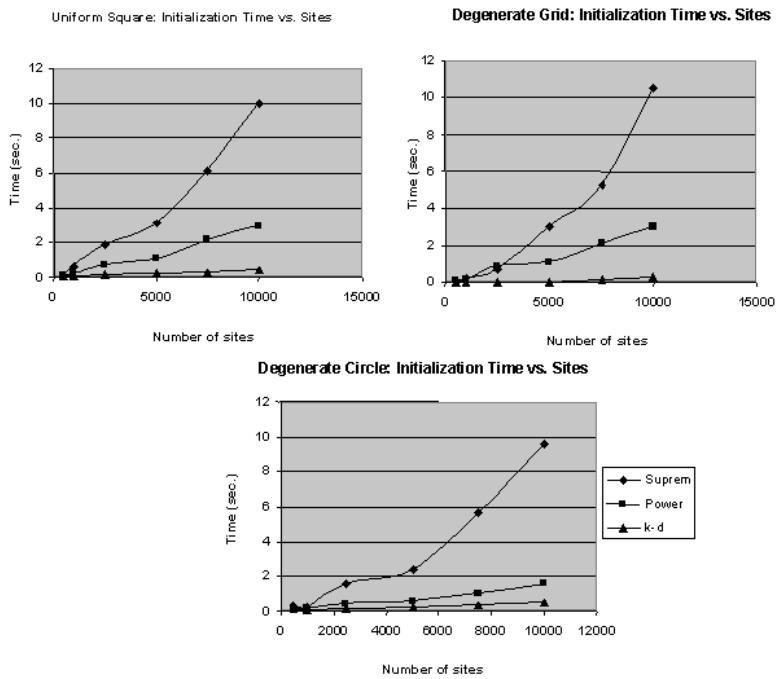


Fig. 4. Time required for building the initial data structure vs. the number of sites

The first series of experiments were performed on randomly generated distributions including uniform square, uniform circle, cross, ring, degenerate grid and degenerate circle distributions. All of the distributions were tested on data sets consisting of 100 to 10000 input sites. The experiments show that the $k-d$ tree method requires much less initialization time than the Voronoi diagram methods, even though the upper bound for both algorithms is $O(n)$. Experimental results demonstrated that the initialization time required to build the data structure is the smallest for the $k-d$ tree based method (see Fig. 4). However, both the power diagram and supremum diagram method consistently outperformed the $k-d$ tree method in terms of the query time required to find the nearest-neighbor (see Fig. 5). This holds for regular as well as close to degenerate configurations. Note that the query time for both VD based methods is very close. The average search length was recorded for all the tests performed, and it exposed a similar linear dependence (growth) as the number of sites increased. Thus, for uniform and degenerate grid distributions it increases from 10 for 100 sites to about 180 for 10000 sites. In the case of circle distribution it increases from 50[DG1] for 100 sites to 5500 for 10000 sites. This result is consistent with the fact that the queries on the circle distribution are usually more time consuming than the queries performed on all other distributions. Based on the results obtained, the

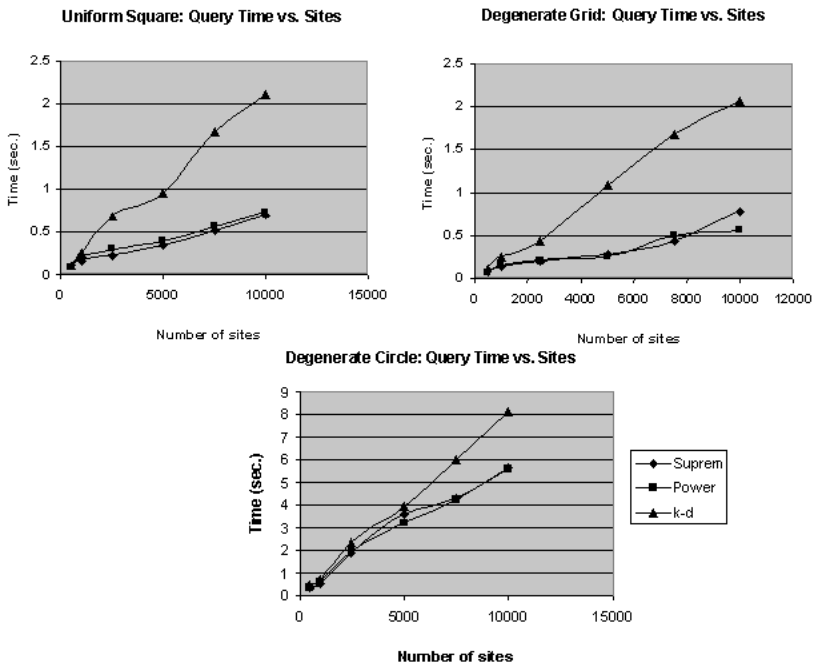


Fig. 5. Time required for performing 1000 queries vs. the number of sites

conclusion can be made that the power and supremum Voronoi diagram method is an efficient data structure for performing nearest-neighbor queries, independent of the site configurations. This was demonstrated for the number of input sites increasing from 100 to 10000. However, the preprocessing time for the VD-based method is quite large compared to the $k-d$ tree method.

Another interesting result is that the VD in either metric can be used for approximate nearest-neighbor searches. The following series of experiments were performed to determine how 'close' the nearest-neighbor found in power metric would be to the nearest-neighbor reported in the supremum metric . The experimental results show that in 95% of all cases the same nearest-neighbor is reported in both metrics, and in 4.5% of remaining cases the two nearest-neighbors reported in different metrics were connected by an edge in the Delaunay triangulation. This shows that it is possible to use either a power diagram or a supremum Voronoi diagram for the *approximate* nearest-neighbor searches.

The third series of the experiments were performed on a data set generated as a result of computer simulation of the granular-type material system for a silo model [7]. The model represents a grain elevator with vertical boundaries and a large number of densely packed grain particles. Test results show that the initialization time for the power diagram method is practically the same as for the

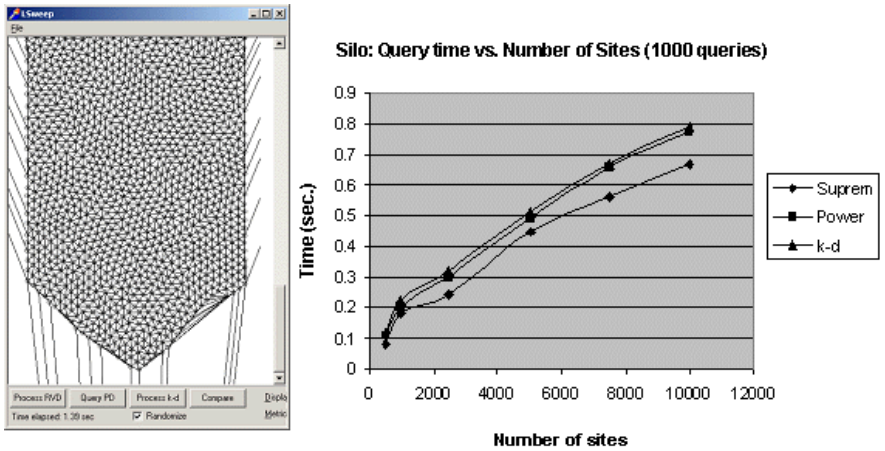


Fig. 6. DT built in power metric for 2500 particles and the running time vs. number of sites.

$k-d$ tree method and the supremum diagram requires significantly more time for initialization. The query time for power diagram is almost the same as for supremum diagram and outperforms the $k-d$ tree method (see Fig. 6).

6 Conclusions

This paper presented an algorithm for an efficient solution of the nearest-neighbor problem for a set of weighted sites based on the generalized Delaunay triangulation. The results obtained clearly demonstrate the applicability of the generalized DT under various distance functions as an efficient, robust and easy to implement method for solving the nearest-neighbor problem. The investigation of the different approaches to select the starting site for the search represents an interesting problem.

Author would like to thank Jon Rokne and Nikolai Medvedev for useful comments and suggestions that helped to improve the paper. Author would also like to express special thanks to Dmitri Gavrilov and Oleg Vinogradov for providing the test data. The work was partly supported by UCRG Research Grant.

References

[1] Aggarwal, P., Raghawan, P. Deferred data structures for the nearest-neighbor problem, Inform. Process. Letters **40** (3) (1991) 119–122.

[2] Bentley J. L. $k-d$ Trees for Semidynamic Point Sets, in Proceedings of the 6th Annual ACM Symposium on Computational Geometry (1990) 187–197.

[3] Berchtold, S., Ertl, B., Keirn, D., Kriegel, H.P., Seidel, T. Fast nearest neighbor search in high-dimensional space, in Proc. of the 14th Intn. Conf. On Data Emg, Orlando, Florida (1998).
[4] Devillers, O., Golin, M., Kedem, K., Schirra, S. Queries on Voronoi Diagrams of Moving Points, Comput. Geom. Theory and Applic. **6** (1996) 315–327.
[5] Edelsbrunner, H., Shah, N. Incremental topological flipping works for regular triangulations, Algorithmica **15** (1996) 223–241.
[6] Friedman, J., Bentley, J., Finkel, R. An Algorithm for Finding Best Matches in Logarithmic Expected Time, ACM Transactions on Mathematical Software, **3**(3) (1977) 209–226.
[7] Gavrilov, D., Vinogradov, O. A cluster in granular systems as a topologically variable structure, in Proc. of 1997 ASCE Symposium on Mechanics of Deformation and Flow of Particulate Materials, Evanston, IL (1997) 299–307.
[8] Gavrilova, M. Proximity and Applications in General Metrics Ph. D. Thesis, Dept. of Computer Science, University of Calgary, Canada (1999).
[9] Gavrilova, M., Rokne, J., Vinogradov O and Gavrilov D. Collision detection algorithms in simulation of granular materials, 1999 ASME Mechanics and Materials Conference, (1999) 283- 284.
[10] Graf, T., Hinrichs, K. A Plane-Sweep Algorithm for the All-Nearest-Neighbors Problem for a Set of Convex Planar Objects, in Proc. 3rd Works. Algm. Data Struct., LNCS, Springer-Verlag **709** (1993) 349–360.
[11] Luchnikov, V.A., Medvedev, N.N., Voloshin, V.P., Geiger, A. Simulation of transport and diffusion of the Voronoi network, in the book: Scientific Computing in Chemical Engineering, Springer-Verlag, Berlin, (1999).
[12] Medvedev, N.N. Voronoi-Delaunay Method for Non-crystalline Structures, SB Russian academy of Science, Novosibirsk (in Russian) (2000).
[13] Murphy, M., Skiena, S. A study of data structures for orthogonal range and nearest neighbor queries in high dimensional spaces, CSE 523/524 Master's Project, Department of Computer Science, SUNYSB (1996).
[14] Okabe, A., Boots, B., Sugihara, K. Spatial Tessellations: Concepts and Applications of Voronoi Diagrams. John Wiley & Sons, Chichester, England (1992).
[15] O'Rourke, J. Computational geometry in C. Cambridge Univ. Press (1994)
[16] Schaudt, B., Drysdale, R. Higher-dimensional Voronoi diagrams for convex distance functions," in Proc. of the 4th Can. Conf. on Comp. Geometry (1992) 274–279.
[17] Shinoda, W., Okazaki, S. A Voronoi analysis of lipid area fluctuation in a bilayer J. Chem. Phys. V. **109**(4) (1998) 1517–1521.

On Dynamic Generalized Voronoi Diagrams in the Euclidean Metric

M.L. Gavrilova and J. Rokne

Department of Computer Science, University of Calgary
Calgary, AB, Canada, T2N 1N4
{marina,rokne}@cpsc.ucalgary.ca

Abstract. The problem of dynamic maintenance of the Voronoi diagram for a set of spheres moving independently in d-dimensional space is addressed in this paper. The maintenance of the generalized Voronoi diagram of spheres, moving alone the given trajectories, requires the calculation of topological events, that occur when $d + 2$ spheres become tangent to a common sphere. The criterion for determination of such a topological event for spheres in the Euclidean metric is presented. This criterion is given in the form of polynomial algebraic equations dependent on the coordinates and trajectories of the moving spheres. These equations are normally solved using numerical methods.

1 Introduction

Such areas as motion planning, computer simulation of physical systems, robotics and computer graphics often deal with geometric objects that move with time [12]. In many applied problems from these areas, collection of geometric objects such as points, disks and spheres, is often considered. Objects can be given along with the analytic functions describing their motion, often specified by polynomials of time. The aim in these problems is to answer questions concerning properties of the system of objects, for instance, finding the closest/furthest pair, predicting the next collision or computing the minimum enclosing circle.

The static Voronoi diagram is often employed by for solving the above problems. Extensive libraries of methods for Voronoi diagram construction, maintenance and querying have been developed over time [14,4]. Weighted Voronoi diagrams for a set of circles in the plane have been introduced and their properties have been studied [2,17,9,8,13]. For the dynamic Voronoi diagram of a set of moving points in the plane only a few algorithms are known. These include computation of the moment in time when the convex hull of algebraically moving points reaches a steady state [1], construction and maintenance of the dynamic Voronoi diagram [7], estimation of upper and lower bounds on the number of combinatorial changes to the diagram over time [15] and solving query problems on the set of moving points [3].

Finally, the problem of construction and maintenance of the dynamic Voronoi diagram for a set of moving objects other than points has rarely been addressed in the literature. Some of the works in this field were devoted to the construction of the dynamic Voronoi diagram for a set of moving hyper-rectangles [11], dynamical maintenance of the Voronoi diagram of line segments in the plane

V.N. Alexandrov et al. (Eds.): ICCS 2001, LNCS 2073, pp. 673–682, 2001.
© Springer-Verlag Berlin Heidelberg 2001

[5] and computation of the time of the topological event in the dynamic Euclidean Voronoi diagram for a set of circles and line segments [10]. In this paper, we extend the result reported in [10] to address the problem of dynamic maintenance of the weighted Euclidean Voronoi diagram for moving spheres in d-dimensional space. An important property of the weighted generalized Voronoi diagram in the Euclidean metric, limiting the number of inscribed spheres, is also established. Based on this result, a criterion to determine the time of the topological event in the Euclidean metric is derived in the form of a system of polynomial algebraic equations.

2 Definitions

Consider a set S of n moving spheres in R^d. Each sphere moves along a given trajectory, described by a function of time. The spheres move in unbounded d-dimensional space. The Voronoi diagram can be used as a data structure to store the topological information about the system of moving objects as well as to answer queries. It is defined as:

Definition 1. The Voronoi diagram for a set of objects S in d-dimensional space is a partitioning of the space into regions, such that each region is the locus of points from S closer to the object $P \in S$ than to any other object $Q \in S$, $Q \neq P$.

The above general definition can be specialized to the set of spheres in the Euclidean metric [14]:

Definition 2. A generalized Euclidean Voronoi diagram (GVD) for a set of sites S in R^d is the set of Voronoi regions $\{\mathbf{x} \in R^d \ \ |d(\mathbf{x}, P) \leq d(\mathbf{x}, Q), \forall Q \in S - \{P\}\}$, where $d(\mathbf{x}, P)$ is the Euclidean distance function between a point $\mathbf{x}$ and a site $P \in S$.

Following the classification of generalized Voronoi diagrams presented in [14] the Euclidean weighted Voronoi diagram is an instance of the class of additively weighted Voronoi diagrams, where $d(\mathbf{x}, P) = d(\mathbf{x}, \mathbf{p}) - r_p$ (see Fig. 1). The distance $d(\mathbf{x}, \mathbf{p})$ between points $\mathbf{x}(x_1, x_2, ..., x_d)$ and $\mathbf{p}(p_1, p_2, ..., p_d)$ in the Euclidean metric is computed as $d(x, y) = \sqrt{\sum_{i=1}^{d} (x_i - y_i)^2}$. According to the definition, the generalized Voronoi region of an additively weighted Voronoi diagram of n sites is obtained as the intersection of $n - 1$ quasi-halfspaces with hyperbolic boundaries. It was shown in [14] that the weighted Euclidean Voronoi diagram for a set of spheres is the set of singly-connected Voronoi regions with the hyperbolic boundaries, star-shaped relative to site P.

The straight-line dual to the Voronoi diagram called a Delaunay tessellation is often used instead of the Voronoi diagram to store topological information for a set of sites:

Definition 3. A generalized Delaunay tessellation corresponding to a generalized Voronoi diagram for a set of spheres S in d-dimensional space is a collection of d-dimensional simplices such that for each generalized Voronoi

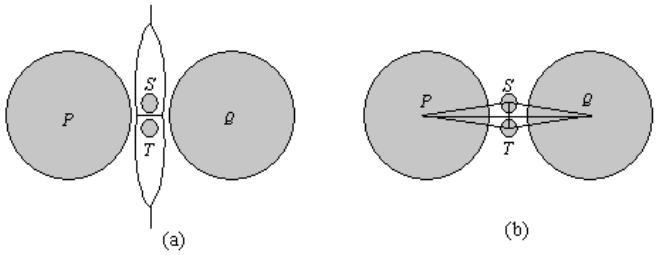


Fig. 1. The Euclidean Voronoi diagram and the corresponding Delaunay triangulation

vertex $\mathbf{v} = EVor(P_1) \cap EVor(P_2) \cap ... \cap EVor(P_{d+1})$ there exists a simplex $(p_1, p_2, \ldots, p_{d+1})$ in the generalized Delaunay tessellation.
Fig. 1 represents the Voronoi diagram in the Euclidean metric and the corresponding Delaunay triangulation for four circles in the plane.

In this paper, we first establish the property that the number of inscribed spheres in the Euclidean metric for $d + 1$ spheres in general positions can be either two, one or zero. Following this, the conditions for the topological swap in the generalized Euclidean Voronoi diagram in d-dimensions are derived.

3 Dynamic Generalized Euclidean VD

3.1 Swap Operation in the Euclidean Metric

Consider the problem of maintaining the ordinary dynamic Voronoi diagram in the plane over some period of time. According to Roos [15], a dynamic Voronoi diagram undergoes two types of changes. The first type is a continuous deformation, where the locations of vertices and lengths of Voronoi edges can change, while proximity relationship between Voronoi sites do not change. The second type of change is the topological change, when Voronoi edges appear and disappear. The discrete moments of time when such change can happen are called topological events.
In order to detect such topological events the Delaunay triangulation is often used. Consider first the ordinary Voronoi diagram. When four moving sites in a quadrilateral, comprising two neighboring triangles of the Delaunay triangulation become cocircular, the corresponding edge of the Voronoi diagram gradually shrinks to zero and then a new edge appears. The corresponding diagonal in a quadrilateral in the Delaunay triangulation is flipped (this operation sometimes called a swap operation) and the future topological events for the newly created quadrilaterals are detected. The conditions for the topological event for Voronoi diagram in the Laguerre geometry and under the Euclidean distance function were established in [13,10]. The d-dimensional swap operation is described in

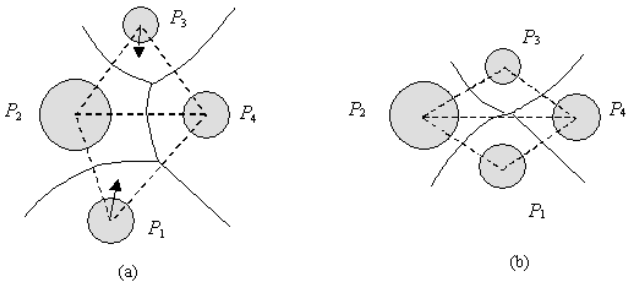


Fig. 2. The Dynamic Euclidean Voronoi diagram transformation

detail in [6,16].

Now, let us consider the topological event in the Euclidean metric. By the dynamic generalized Euclidean Voronoi diagram (referred to as the Euclidean Voronoi diagram in sequel) we mean the generalized Voronoi diagram in the Euclidean metric for a set of sites moving independently along the given trajectories. The topological event in the Euclidean metric can be illustrated on the following example. Consider a case when two circles P_1 and P_3 move towards each other along the straight-line trajectory in the direction shown by arrows (see Fig. 2**(a)**). Assume for simplicity that other two circles P_2 and P_4 remain in their spatial positions. At some moment of time t four circles become cocircular and the edge between sites P_2 and P_4 is reduced to zero. As the circles P_1 and P_3 continue to move toward each other, the new edge between sites P_1 and P_3 appears and its length increases with time (Fig. 2**(b)**). The topology of the Euclidean Voronoi diagram changes. The corresponding changes in the Euclidean Delaunay triangulation are shown in Fig. 2 by dashed lines.

The following conclusions can be drawn from the above discussion. First, for a finite set of sites S the topological structure of the Euclidean Voronoi diagram is locally stable, i.e. only continuous deformations take place, under sufficiently small continuous motion of sites. Secondly, the topological changes in the structure of the Euclidean Voronoi diagram are characterized by swaps of adjacent triangles (tetrahedrons) in the Delaunay triangulation (tessellation).

3.2 Dynamic Euclidean Voronoi Diagram Maintenance

The algorithm for maintenance of the Voronoi diagram for n circles, presented in [10], is now extended to handle the d-dimensional case.

1. (Preprocessing) Construct the Delaunay tessellation for the original site distribution. For every existing d-dimensional quadrilateral (quadrilateral formed by $d+2$ neighboring spheres) in the Delaunay tessellation calculate

the next topological event. Insert all such events into the event priority queue sorted according to the time order.

2. (Iteration) Take the next topological event from the event queue. Update the Delaunat tessellation corresponding to the d-dimensional swap operation

3. Delete all topological events from the event queue, planned for no longer existing d-dimensional quadrilaterals.

4. Compute the new topological events for all new d-dimensional quadrilaterals and insert them into the event queue.

Preprocessing step requires $O(n^{\lceil d/2 \rceil + 1})$ time (see [6,16], for example). The swap operation takes $O(1)$ time, insertion to the queue in sorted order requires $O(\log n)$ time, and deletion from the queue takes $O(1)$ time when for each d-dimensional quadrilateral we store pointers on the events scheduled for this quadrilateral. The maximum size of the queue is $O(n^{\lceil d/2 \rceil})$, since there is at most one event scheduled at any moment of time for each quadrilateral. The space required to store the tessellation is $O(n^{\lceil d/2 \rceil})$. The total number of topological events depends on the trajectories of the moving sites and the elapsed time (upper bounds estimates were obtained for certain types of trajectories in [15]). The above is now summarized.

Lemma 1. The algorithm for maintenance of the Voronoi diagram for a set of sites in plane takes $O(n^{\lceil d/2 \rceil + 1})$ preprocessing time, $O(n^{\lceil d/2 \rceil})$ space and each topological event uses $O(d \log n)$ time.

To determine the time of the topological event, or the moment when $d + 2$ Voronoi sites are co-spherical, it is required to find the minimal root t_0 of the equation $INCIRCLE(P_1, P_2, ..., P_{d+2}) = 0$. In the above, $P_i = P_i(t)$, $i = 1..d + 2$ are the coordinates of the $d + 2$ moving spheres, where t denotes the time. The form of the $INCIRCLE$ function depends on the metric being used. If the value of the $INCIRCLE$ function is positive then the empty-sphere condition is satisfied, when it is equal to zero, then the topological event occurs. It was shown that in the planar case the $INCIRCLE$ function in power metric can be computed as a 4×4 determinant [9] and that in Euclidean metric it can be represented as a 6th degree polynomial of time [10]. Now, let us consider the generalization to d-dimensions under the Euclidean metric.

3.3 The Number of Inscribed Spheres in the Euclidean Metric

Let $P_i = \{\mathbf{p}_i = (x_{i1}, x_{i2}, ..., x_{id}), r_i\}, i = 1..d+1$ be $d+1$ spheres in d-dimensional space. We will show how to obtain the coordinates and the radius of an inscribed sphere $C = \{\xi = (\xi_1, \xi_2, ..., \xi_d), \rho\}$ and establish the number of such spheres. Let us first formally define the sphere inscribed among the $d + 1$ spheres in the d-dimensional space.

Definition 4. A sphere $C = \{\xi, \rho\}$ inscribed among $d+1$ spheres $P_1, P_2, ..., P_{d+1}$ is a sphere with center $\xi = (\xi_1, \xi_2, ..., \xi_d)$ and radius ρ, such that $\rho = d(\xi, P_1) = d(\xi, P_2) = ... = d(\xi, P_{d+1})$.

Now, let us reduce the values of the radii of the $d+1$ spheres by the radius of the smallest sphere. Without loss of generality assume that the smallest sphere has index $d+1$. Let us define a coordinate system with the center of coordinates at point $\mathbf{p}_{d+1}$. Then the transformed coordinates of the given spheres are $P_i^* = \{\mathbf{p}_i^* = (x_{i1}^*, x_{i2}^*, ..., x_{id}^*), r_i^*\}, i = 1..d$, where $x_{ij}^* = x_{ij} - x_{d+1,j}, i, j = 1..d$, and $r_i^* = r_i - r_{d+1}$. The last sphere is transformed into a point at the origin of coordinates. The unknown inscribed sphere coordinates will change to $\xi_j^* = \xi_j - x_{d+1,j}, j = 1..d$, and $\rho^* = \rho + r_{d+1}$.

We will use the fact that the coordinates of the inscribed sphere satisfy the equations

$$d(\xi^*, \mathbf{p}_i^*) = \rho^* + r_i^*, i = 1..d+1. \tag{1}$$

Expanding the distance function, we get:

$$\begin{cases} (x_{i1}^* - \xi_1^*)^2 + (x_{i2}^* - \xi_2^*)^2 + ... + (x_{id}^* - \xi_d^*)^2 = (\rho^* + r_i^*)^2, i = 1..d \\ (\xi_1^*)^2 + (\xi_2^*)^2 + ... + (\xi_d^*)^2 = (\rho^*)^2 \end{cases} \tag{2}$$

The last equation can be subtracted from the remaining equations to cancel the quadratic terms

$$\begin{cases} 2x_{i1}^*\xi_1^* + 2x_{i2}^*\xi_2^* + ... + 2x_{id}^*\xi_d^* + 2\rho^* r_i^* = w_i^*, i = 1..d \\ (\xi_1^*)^2 + (\xi_2^*)^2 + ... + (\xi_d^*)^2 = (\rho^*)^2 \end{cases} \tag{3}$$

where $w_i^* = (x_{i1}^*)^2 + (x_{i2}^*)^2 + ... + (x_{id}^*)^2 - (r_i^*)^2, i = 1..d$.

The solution for this system can be obtained by the following steps. The first d equations are linear in $(\xi_1, \xi_2 ..., \xi_d, \rho)$. This linear system has d equations and $d+1$ variables. Denote the matrix of the system by $\mathbf{A}$, the column of unknowns by $\mathbf{x}$ and the right-hand side column by $\mathbf{b}$.

The following three cases are possible:

Case 1. $rank(\mathbf{A}) = rank(\mathbf{A}|\mathbf{b}) = d$.

This is the general case. The linear system can be resolved for d of the variables leaving one of the variables as a free parameter. To determine which variable is left free, a variable that can be moved into the right-hand side of the system must be found so that the determinant of the remaining system is non-zero. Thus, a non-zero $[d \times d]$ minor of matrix $\mathbf{A}$ must be found. Assume that ξ_k^* is left as a free variable. The remaining unknowns will all be linear functions of ξ_k^*. They can be substituted into the last equation, which turns into a quadratic equation for ξ_k^*. It can have two, one or no real solutions. Consequently, the following statement is true:

Lemma 2. The number of inscribed spheres in the Euclidean metric for a given $d+1$ spheres in d-dimensional space in general positions (i.e. $rank(\mathbf{A}) = d$) can be either two, one or zero.

Note that even though the system can have up to two solutions, only those where the radius of the inscribed sphere is positive should be selected. Also

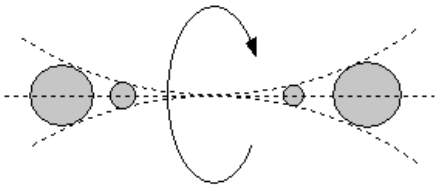


Fig. 3. Infinite number of inscribed spheres.

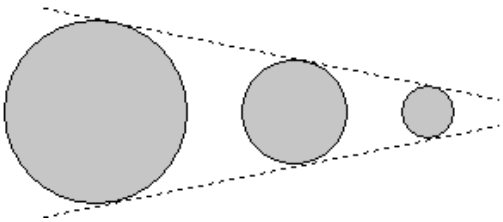


Fig. 4. Linearly dependent spheres.

note that each of the inscribed spheres corresponds to a distinct Delaunay tetrahedron in the Delaunay tessellation.
Case 2. $rank(\mathbf{A}) = rank(\mathbf{A}|\mathbf{b}) < d$.
In this case, the linear system has an infinite number of solutions, and, consequently, infinitely many inscribed spheres. An example of such a system is given in Fig. 3.
Case 3. $rank(\mathbf{A}) < rank(\mathbf{A}|\mathbf{b}) \leq d$.
In this case, the linear system has no solutions, and consequently, there are no inscribed spheres. An example of such a system is presented in Fig. 4.
Note that cases 2 and 3 both represent degenerate sphere arrangements, because the spheres are linearly dependent when $rank(\mathbf{A}) < d$.

3.4 Topological Event Computation

Now, let us obtain the formulas to compute the topological event in the Euclidean tessellation of moving sites. Let $P_i = \{(x_i = x_i(t), y_i = y_i(t)), r_i\}$, $i = 1..d+2$ be a set of spheres with centers $(x_i(t), y_i(t))$ given by analytic functions of time and radii r_i.
Theorem 1. The time of the topological event in a Delaunay d-dimensional quadrilateral of $d+2$ spheres $P_i = \{(x_i = x_i(t), y_i = y_i(t)), r_i\}$, can be found as the minimum real root t_0 of the equation

$$A_1^2 + A_2^2 + ... + A_d^2 = A_{d+1}^2, \tag{4}$$

where

$$A = \begin{vmatrix} x_{11} - x_{d+2,1} & x_{12} - x_{d+2,2} & ... & x_{1d} - x_{d+2,d} & r_1 - r_{d+2} \\ x_{22} - x_{d+2,1} & x_{22} - x_{d+2,2} & ... & x_{2d} - x_{d+2,d} & r_2 - r_{d+2} \\ ... & ... & ... & ... & ... \\ x_{d+1,1} - x_{d+2,1} & x_{d+1,2} - x_{d+2,2} & ... & x_{d+1,d} - x_{d+2,d} & r_{d+1} - r_{d+2}, \end{vmatrix}$$

and A_i is obtained by replacing the i-th column of A by the column $\left[w_j^*\right]_{j=1}^{d+1}$, where$w_i = \sum_{j=1}^{d} \left(x_{ij} - x_{d+2,j}\right)^2 - \left(r_i - r_{d+2}\right)^2, i = 1..d+1.$

Proof.
The $INCIRCLE$ function for $d+2$ spheres can be obtained by determining the coordinates of the inscribed sphere(s) for the first $d+1$ spheres, and then computing the distance from the last sphere to the inscribed sphere(s): $INCIRCLE(P_1, ..., P_{d+2}) = \rho - d(\xi, P_{d+2})$. In the Euclidean metric, this formula is transformed to:

$$INCIRCLE\left(P_1, P_2, ..., P_{d+2}\right) = \rho - \left(\sqrt{\sum_{i=1}^{d} \left(\xi_i - x_{d+2,i}\right)^2} - r_{d+2}\right), \qquad (5)$$

where $(\xi_1, \xi_2 ..., \xi_d, \rho)$ are the coordinates of the sphere inscribed among the spheres $P_1, P_2, ..., P_{d+1}$.
The condition can be rewritten as the following system of equations:

$$d\left(\mathbf{p}_i, \xi\right) = r_i + \rho, i = 1..d+2. \qquad (6)$$

Performing transformations similar to those described above reduces the radii of all spheres by the radius of the smallest sphere (assume that this is the $(d+2)$-nd sphere). The origin of the coordinates is moved to the center of the smallest sphere. Denote $x_{ij}^* = x_{ij} - x_{d+2,j}$ and ${d_i}^* = d_i - d_{d+2}$.
Then the second-degree terms in the first $d+1$ equations can be cancelled

$$\begin{cases} 2x_{i1}^*\xi_1^* + 2x_{i2}^*\xi_2^* + ... + 2x_{id}^*\xi_d^* + 2\rho^* r_i^* = w_i^*, i = 1..d+1 \\ \left(\xi_1^*\right)^2 + \left(\xi_2^*\right)^2 + ... + \left(\xi_d^*\right)^2 = \left(\rho^*\right)^2 \end{cases} \qquad (7)$$

In the above system, ${w_i}^* = \left(x_{i1}^*\right)^2 + \left(x_{i2}^*\right)^2 + ... + \left(x_{id}^*\right)^2 - \left(r_i^*\right)^2, i = 1..d+1$.
The first $d+1$ equations represent a linear system

$$2\begin{bmatrix} x_{11}^* & x_{12}^* & ... & x_{1d}^* & r_1^* \\ x_{21}^* & x_{22}^* & ... & x_{2d}^* & r_2^* \\ ... & ... & ... & ... & ... \\ x_{d+1,1}^* & x_{d+1,2}^* & ... & x_{d+1,d}^* & r_{d+1}^* \end{bmatrix} \begin{bmatrix} \xi_1^* \\ ... \\ \xi_d^* \\ \rho^* \end{bmatrix} = \begin{bmatrix} w_1^* \\ w_2^* \\ ... \\ w_{d+1}^* \end{bmatrix} \qquad (8)$$

Assuming that the determinant of the linear system is non-zero, the system will always have a unique solution. The formulas for the center and the radii of the

sphere inscribed among the $d+2$ spheres with the modified radii can be explicitly written using Cramer's rule:

$$\xi_i^* = \frac{1}{2}\frac{A_i}{A}, i = 1..d; \rho^* = \frac{1}{2}\frac{A_{d+1}}{A}, \tag{9}$$

where, returning to the original coordinates,

$$A = \begin{vmatrix} x_{11} - x_{d+2,1} & x_{12} - x_{d+2,2} & ... & x_{1d} - x_{d+2,d} & r_1 - r_{d+2} \\ x_{22} - x_{d+2,1} & x_{22} - x_{d+2,2} & ... & x_{2d} - x_{d+2,d} & r_2 - r_{d+2} \\ ... & ... & ... & ... & ... \\ x_{d+1,1} - x_{d+2,1} & x_{d+1,2} - x_{d+2,2} & ... & x_{d+1,d} - x_{d+2,d} & r_{d+1} - r_{d+2}, \end{vmatrix}$$

and A_i is obtained by replacing the i-th column of A by the column $\left[w_j^*\right]_{j=1}^{d+1}$

$w_i = \sum_{j=1}^{d} \left(x_{ij} - x_{d+2,j}\right)^2 - \left(r_i - r_{d+2}\right)^2, i = 1..d+1.$

Then the formulas for the center and radii of the inscribed sphere are substituted into the last quadratic equation from the system (7), arriving at the condition

$$A_1^2 + A_2^2 + ... + A_d^2 = A_{d+1}^2. \tag{10}$$

The theorem is now proven.

The additional condition, requiring that the radius of the inscribed sphere must be positive, must be imposed as well.

$$\rho = \frac{1}{2}\frac{A_{d+1}}{A} - r_{d+2} > 0. \tag{11}$$

Note that all coordinates are analytical functions of time. When the spheres move by straight-line trajectories, the condition turns into an equation which is an 8th degree polynomial of time. As the spheres move with time, (10) and (11) can be written as

$$f(t) = 0 \tag{12}$$

$$g(t) > 0 \tag{13}$$

Note that the function $f(t)$ is in general non-zero. The first t_0 satisfying $f(t_0) = 0$ and the condition (13) represents the first topological event encountered. The complexity of the solution of (12) and (13) for t_0 clearly depends on the nature of the functions describing the movements of the circles. Even if these functions are linear, solving $f(t) = 0$ reduces to finding the zeros of a high degree polynomial of time. This will require an iterative numerical method, for example Newton's method.

4 Conclusion

Criterion for determination of the time of topological event in a Voronoi diagram for moving spheres in d-dimensions has been presented. The results are given in an algebraic form and can be applied to compute the dynamic generalized Voronoi diagram in the Euclidean metric.

References

1. Atallah, M., Some dynamic computational geometry problems, Computers and Mathematics with Applications, **11** (1985) 1171–1181.
2. Aurenhammer, F. "Voronoi diagrams - A survey of a fundamental geometric data structure," ACM Computing Surveys, **23(3)** (1991) 345–405.
3. Devillers, O., Golin, M., Kedem, K. and Schirra, S. "Revenge of the dog: queries on Voronoi diagrams of moving points," in Proc. of the 6th Canadian Conference on Computational Geometry, (1994) 122–127.
4. Dey, T.K., Sugihara K. and Bajaj, C. L. DT in three dimensions with finite precision arithmetic, Comp. Aid. Geom. Des **9**(1992) 457-470
5. Dobrindt, K. and Yvinec, M. Remembering conflicts in history yields dynamic algorithms, in Proceedings of the 4th International Symposium on Algorithms and Computation (1993) 21–30.
6. Edelsbrunner, H. and Shah, N. Incremental topological flipping works for regular triangulations, Algorithmica, **15**, (1996) 223-241
7. Fu, J. and Lee, R. Voronoi diagrams of moving points in the plane, Int. J.Comp.Geom.& Appl., **1(1)** (1991) 23-32.
8. Gavrilova, M. Robust algorithm for finding nearest-neighbors under L-1, L-inf and power metrics in the plane, to appear in the Proceedings of the int. Conf. on Comp. Sciences 2001, San Francisco, USA (2001)
9. Gavrilova, M. and Rokne, J. An Efficient Algorithm for Construction of the Power Diagram from the Voronoi Diagram in the Plane, Intern. Jour. of Computer Math., Overseas Publishers Association, **61** (1997) 49–61.
10. Gavrilova,M. and Rokne,J. Swap conditions for dynamic VD for circles and line segments, Comp-Aid. Geom.Design **16** (1999) 89–106.
11. Gupta, P., Janardan, R. and Smid, M., Fast algorithms for collision and proximity problems involving moving geometric objects, Report MPI-I-94-113, Max-Planck-Insitut fur Informatik, Zaarbrucken (1994).
12. Hubbard, P. Approximating polyhedra with spheres for time-critical collision detection, ACM Transaction on Graphics, **15(3)** (1996) 179-210.
13. Kim, D.-S., Kim, D. Sugihara, K. and Ryu, J. Most Robust Algorithm for a Circle Set Voronoi Diagram in a Plane, to appear in the Proc. of the Int. Conf. on Comp. Sciences'01, San Francisco, USA (2001)
14. Okabe, A., Boots, B. and Sugihara, K. Spatial tessellations: concepts and applications of Voronoi diagrams,John Wiley and Sons, Chichester, West Sussex, England, (1992) 205–208.
15. Roos, T. Voronoi diagrams over dynamic scenes, Discrete Appl. Mathem., Netherlands, **43(3)** (1993) 243–259.
16. Schaudt, B. and Drysdale, R. Higher-dimensional Voronoi diagrams for convex distance functions, in Proceedings of the 4th Canadian Conference on Computational Geometry, (1992) 274–279.
17. Sugihara, K. Approximation of generalized Voronoi Diagrams by ordinary Voronoi diagrams, CVGIP: Graph. Models Image Process, **55** (1993) 522–531.

Computing Optimal Hatching Directions in Layered Manufacturing*

Man Chung Hon[1], Ravi Janardan[1], Jörg Schwerdt[2], and Michiel Smid[2]

[1] Dept. of Computer Science & Engineering, University of Minnesota, Minneapolis, MN 55455, U.S.A.
{hon,janardan}@cs.umn.edu
[2] Fakultät für Informatik, Otto-von-Guericke-Universität Magdeburg, D-39106 Magdeburg, Germany.
{schwerdt,michiel}@isg.cs.uni-magdeburg.de

Abstract. In Layered Manufacturing, a three-dimensional polyhedral solid is built as a stack of two-dimensional slices. Each slice (a polygon) is built by filling its interior with a sequence of parallel line segments, of small non-zero width, in a process called hatching. A critical step in hatching is choosing a direction which minimizes the number of segments. Exact and approximation algorithms are given here for this problem, and their performance is analyzed both experimentally and analytically. Extensions to several related problems are discussed briefly.

1 Introduction

This paper addresses a geometric problem motivated by *Layered Manufacturing* (LM), which is an emerging technology that allows the construction of physical prototypes of three-dimensional parts directly from their computer representations, using a "3D printer" attached to a personal computer.

The basic idea behind LM is very simple. A direction is first chosen to orient the computer model suitably. The model is then sliced with a set of equally spaced horizontal planes, resulting in a stack of 2-dimensional polygons. Starting from the bottom, each slice is sent to the LM machine and built on top of the layers below it. There are several different ways how this process is carried out physically. One particular implementation is through a process called *Stereolithography* [3]. Here the model is built in a vat of liquid which hardens when exposed to light. A laser is used to trace the boundary of each slice and then fill in its interior via a series of parallel line segments (Fig. 1(a)); this process is called *hatching*. Another process called Fused Deposition Modeling hatches the slices by depositing fine strands of molten plastic via a nozzle.

The hatching process in LM influences the process cost and build time quite significantly. For instance, in Stereolithography, the number of times the laser's

* Research of MCH and RJ supported, in part, by NSF grant CCR–9712226. Portions of this work were done when RJ visited the University of Magdeburg and JS and MS visited the University of Minnesota under a joint grant for international research from NSF and DAAD.

V.N. Alexandrov et al. (Eds.): ICCS 2001, LNCS 2073, pp. 683–692, 2001.
© Springer-Verlag Berlin Heidelberg 2001

path hits the slice boundary is proportional to the number of line segments. It is important to keep this quantity small since it determines the number of times the laser has to decelerate and stop, change directions, and then accelerate; frequent starts and stops are time-consuming and reduce the life of the laser. The number of line segments can be kept small by picking a suitable hatching direction. We define this problem formally in the next section.

1.1 The Hatching Problem and Its Approximation

A slice is a simple polygon $\mathcal{P}$, possibly with holes, in the 2-dimensional plane. Let $\mathbf{d}$ be a unit vector in the plane, and $\ell_0(\mathbf{d})$ the line through the origin with direction $\mathbf{d}$; $\mathbf{d}$ is the *hatching direction*. Let $\mathcal{L}(\mathbf{d})$ be the set of all lines that are parallel to $\ell_0(\mathbf{d})$ and whose distances to $\ell_0(\mathbf{d})$ are multiples of δ, the width of the path. We denote by $\mathcal{S}_\ell$ the set containing the line segments in the intersection between ℓ and $\mathcal{P}$, and define $H(\mathbf{d}) := \sum_{\ell \in \mathcal{L}(\mathbf{d})} |\mathcal{S}_\ell|$. (Fig. 1(b).) The optimization problem can be stated formally as follows:

Problem 1 (Hatching Problem). Given a simple n-vertex polygon $\mathcal{P}$, possibly with holes, compute a hatching direction $\mathbf{d}$ such that $H(\mathbf{d})$ is minimized.

Suppose the width δ of the tool-tip is infinitesimally small. (By "tool" we mean, e.g., the laser in Stereolithography or the nozzle in Fused Deposition Modeling.) Then, given any hatching direction $\mathbf{d}$, the number of times the hatching path runs into an edge e of $\mathcal{P}$ is proportional to the length of e's projection *perpendicular* to $\mathbf{d}$. Thus the solution to the hatching problem can be approximated by finding a direction which minimizes the total length of the projections of the edges of $\mathcal{P}$ onto a line perpendicular to this direction. (Clearly the smaller δ is, the better is the approximation.) This yields the following problem, where, for simplicity, we consider not the edges themselves but their outward normals, each with the same length as its corresponding edge and translated to the origin.

Problem 2 (Projection Problem). Given a finite set $\mathcal{S}$ of n vectors in the plane, each beginning at the origin, find a unit vector $\mathbf{d}$ such that $\sum_{\mathbf{v} \in \mathcal{S}} |\mathbf{v} \cdot \mathbf{d}|$ is minimized.

Note that Problem 2 depends only on the lengths and orientations of the edges of the original polygon, and not on how they connect to each other in the polygon. This suggests that we can find a *globally* optimal hatching direction for *all* the layers by projecting the edges from all layers onto the xy-plane and running our algorithm on the resulting set of vectors.

1.2 Contributions

In Sections 2 and 3 we present two simple and efficient algorithms for Problem 2; this yields an approximation to the optimal hatching direction. For comparison, we also designed an algorithm for Problem 1 which computes an optimal hatching direction; this algorithm is more complex and is described in Section 4. We

establish the performance of the approximation algorithms in two ways: First, we implemented the algorithms of Sections 3 and 4 and tested them on real-world polyhedral models obtained from industry (Section 5). We discovered that the approximation algorithm works very well in practice. Second, we show that, under reasonable assumptions, the number of hatching segments produced by the approximation algorithms is only a constant times more than the number produced by the optimal algorithm (Section 6). In Section 7, we discuss applications of the approximation algorithms to other related problems. For lack of space, we omit many details here; these can be found in [1,5].

2 Minimizing the Projected Length of a Simple Polygon

Recall what we want to accomplish. We are given a simple polygon, from which we get a set $\mathcal{S}$ of outward-pointing normal vectors $\mathbf{n}_e$ for each edge e, with $\mathbf{n}_e$ having the same length as e and beginning at the origin. We want to compute a direction $\mathbf{d}$ that minimizes the sum $\sum_e |\mathbf{n}_e \cdot \mathbf{d}|$. We replace all the vectors in $\mathcal{S}$ that point in the same direction by their sum. We then sort the vectors in circular order and do a circular walk around the origin. We keep an initially empty chain of vectors during our walk. Whenever we encounter a vector in $\mathcal{S}$, we put it onto the chain, with its tail at the head of the old chain.

It is easy to see that the sum of all these normals $\sum_e \mathbf{n}_e$ is *zero*, since our polygon is closed. It follows that we will get a polygon at the end of our circular walk. Moreover, this polygon is convex because the vectors are added in sorted order. Now it is clear that, for any direction $\mathbf{d}$, the sum of the absolute values of the dot products of the vectors in $\mathcal{S}$ w.r.t. $\mathbf{d}$ is twice the width of this convex polygon in the direction perpendicular to $\mathbf{d}$. (Fig. 2). Therefore, finding the minimizing direction in Problem 2 is equivalent to finding the direction that minimizes the width of the convex polygon. Using any of the standard algorithms that compute the smallest width of a convex polygon [2], we have:

Theorem 1. *Given a simple n-vertex polygon $\mathcal{P}$ in the plane, we can compute in $O(n \log n)$ time and using $O(n)$ space a unit vector $\mathbf{d}$ such that the sum $\sum_e |\mathbf{n}_e \cdot \mathbf{d}|$ is minimized.*

As noted in the discussion leading up to Problem 2, the direction $\mathbf{d}$ in Theorem 1 can be used as an approximation to the optimal hatching direction sought in Problem 1. A similar algorithm was discovered independently in [4].

3 An Alternative Algorithm

In this section, we present another approach to Problem 2. This algorithm has the advantage that it works on any set of vectors, not just those corresponding to the edge normals of a simple polygon; moreover, it generalizes easily to higher dimensions.

Consider the set $\mathcal{S}$ of normal vectors in the plane, each with its tail at the origin. We pick an arbitrary vector $\mathbf{d}$ as a candidate direction and draw a line perpendicular to $\mathbf{d}$ through the origin. This line cuts the plane into two half-planes. The normals that lie in the opposite half-plane as $\mathbf{d}$ will register a negative value in their inner products with $\mathbf{d}$. We correct the inner products of these vectors with a minus sign. This corresponds to "reflecting" these vectors through the origin. We replace the downward-pointing vectors (w.r.t. $\mathbf{d}$) with their reflected copies (Fig. 3). We call this new set of vectors $\tilde{\mathcal{S}}$.

All the vectors $\tilde{\mathbf{v}}$ in $\tilde{\mathcal{S}}$ lie in the same closed half-plane as $\mathbf{d}$. Therefore $\sum_{\mathbf{v}\in\mathcal{S}} |\mathbf{v}\cdot\mathbf{d}| = \sum_{\tilde{\mathbf{v}}\in\tilde{\mathcal{S}}} (\tilde{\mathbf{v}}\cdot\mathbf{d}) = (\sum_{\tilde{\mathcal{S}}} \tilde{\mathbf{v}})\cdot\mathbf{d}$. In other words, the sum of all the projection lengths is equal to the inner product of $\mathbf{d}$ with a single vector $\sum_{\tilde{\mathcal{S}}} \tilde{\mathbf{v}}$. If no element of $\tilde{\mathcal{S}}$ is on the cutting line, nothing prevents us from rotating $\mathbf{d}$ away from $\sum_{\tilde{\mathcal{S}}} \tilde{\mathbf{v}}$ and in the process decreasing the inner product it makes with $\sum_{\tilde{\mathcal{S}}} \tilde{\mathbf{v}}$. We can keep doing this until one of the vectors $\tilde{\mathbf{v}}$ is on the cutting line. Now any further movement of $\mathbf{d}$ will cause $\tilde{\mathbf{v}}$ to go to the other side of the cutting line and cause the total projection length to increase. Thus, the position of the cutting line that coincides with one of the input vectors must be a local minimum for the total projected length.

We can update $\sum_{\tilde{\mathcal{S}}} \tilde{\mathbf{v}}$ efficiently if we visit the vectors in a circular order. Specifically, each vector $\tilde{\mathbf{v}}$ has associated with it two regions, separated by the line perpendicular to $\tilde{\mathbf{v}}$. In our walk, whenever we pass this line, we know that the associated vector's contribution to the sum changes sign. If $\tilde{\mathbf{v}}_i$ is the associated vector, we subtract $2\tilde{\mathbf{v}}_i$ from $\sum_{\tilde{\mathcal{S}}} \tilde{\mathbf{v}}$, one copy to take it off from the sum, and another copy to insert it back in with a negative sign. We use the newly updated vector sum to calculate the projection at that event point. Since the update can be done in $O(1)$ time, we get the same result as in Theorem 1.

4 An Exact Algorithm for the Hatching Problem

In this section, we give an outline of our algorithm that solves Problem 1. W.l.o.g., we may assume that no vertex of the polygon $\mathcal{P}$ is at the origin and that no three successive vertices of $\mathcal{P}$ are collinear.

Since $H(\mathbf{d}) = H(-\mathbf{d})$ for any direction $\mathbf{d}$, it suffices to compute an optimal hatching direction $\mathbf{d} = (d_1, d_2)$ for which $d_2 \geq 0$. The idea of our algorithm is as follows. We start with an initial direction $\mathbf{d} = (-1, 0)$, and rotate it in clockwise order by an angle of π until $\mathbf{d} = (1, 0)$. At certain directions $\mathbf{d}$, the value of $H(\mathbf{d})$ changes. We will call such directions *critical*. During the rotation, we update the value of $H(\mathbf{d})$ at each such critical direction.

During the rotation, the collection $\mathcal{L}(\mathbf{d})$ rotates, with the origin being the center of rotation. We give necessary conditions for a direction $\mathbf{d}$ to be critical. There are two types of directions $\mathbf{d}$, for which $H(\mathbf{d})$ changes.

Type 1: The subset of lines in $\mathcal{L}(\mathbf{d})$ that intersect the polygon $\mathcal{P}$ changes.

We analyze when this can happen. Let $CH(\mathcal{P})$ be the convex hull of $\mathcal{P}$. Note that any line intersects $\mathcal{P}$ if and only if it intersects $CH(\mathcal{P})$. Let $\mathbf{d}$ be a direction at which the subset of $\mathcal{L}(\mathbf{d})$ that intersects $\mathcal{P}$ changes. Let $\mathbf{d}^{\perp}$ be a direction

that is orthogonal to $\mathbf{d}$. Then there must be a vertex v on $CH(\mathcal{P})$ such that: (i) v is extreme in one of the directions $\mathbf{d}^\perp$ and $-\mathbf{d}^\perp$, and (ii) v lies on a line of $\mathcal{L}(\mathbf{d})$, i.e., the distance between v and the line $\ell_0(\mathbf{d})$ through the origin having direction $\mathbf{d}$, is a multiple of δ.

Type 2: For some line $\ell \in \mathcal{L}(\mathbf{d})$, the set $\mathcal{S}_\ell$ of line segments (of positive length) in the intersection $\ell \cap \mathcal{P}$ changes.

If this happens, then there is a vertex v of $\mathcal{P}$ such that: (i) v lies on a line of $\mathcal{L}(\mathbf{d})$, i.e., the distance between v and the line $\ell_0(\mathbf{d})$ is a multiple of δ, and (ii) both vertices of $\mathcal{P}$ that are adjacent to v are on the same side of the line $\ell_v(\mathbf{d})$ through v that is parallel to $\ell_0(\mathbf{d})$. (We have to be careful with degenerate cases.)

Let $\mathbf{D}$ be the set of all directions $\mathbf{d}$ for which there is a vertex v of $\mathcal{P}$ whose distance to the line $\ell_0(\mathbf{d})$ is a multiple of δ. It follows from above that $\mathbf{D}$ contains all critical directions. We now give a brief overview of the algorithm.

Step 1: For each vertex v of $\mathcal{P}$, compute all directions $\mathbf{d} = (d_1, d_2)$ for which $d_2 \geq 0$, and for which the distance between v and the line $\ell_0(\mathbf{d})$ is a multiple of δ. Let $\mathbf{D}$ be the resulting set of directions.

A simple geometric analysis shows that this step can be reduced to solving $2(1 + \|v\|/\delta)$ quadratic equations for each vertex v of $\mathcal{P}$. Hence, the time for Step 1 is $O(|\mathbf{D}|)$, where $|\mathbf{D}| \leq 2n(1 + \max_v \|v\|/\delta)$.

Step 2: Sort the directions of $\mathbf{D}$ in the order in which they are visited when we rotate the unit-vector $(-1, 0)$ by an angle of π in clockwise order. We denote this ordering relation by $\prec$. The time for this step is $O(|\mathbf{D}| \log |\mathbf{D}|)$.

Let m be the number of distinct directions in the set $\mathbf{D}$. We denote the sorted elements of $\mathbf{D}$ by $\mathbf{d}^0 \prec \mathbf{d}^1 \prec \ldots \prec \mathbf{d}^{m-1}$. Note that for any i and any two directions $\mathbf{d}$ and $\mathbf{d}'$ strictly between $\mathbf{d}^i$ and $\mathbf{d}^{i+1}$, we have $H(\mathbf{d}) = H(\mathbf{d}')$.

Step 3: Let $\mathbf{d}_s$ be a direction that is not in $\mathbf{D}$. Compute $H(\mathbf{d}_s)$ for this direction.

Recall that $H(\mathbf{d}_s)$ is the number of line segments of positive length in the intersection of $\mathcal{P}$ with $\mathcal{L}(\mathbf{d}_s)$. The endpoints of any such line segment are on the boundary of $\mathcal{P}$. Hence, the total number of intersection points between $\mathcal{P}$ and the lines in $\mathcal{L}(\mathbf{d}_s)$ is twice $H(\mathbf{d}_s)$. For any edge $e = (u, v)$ of $\mathcal{P}$, let I_e be the number of lines in $\mathcal{L}(\mathbf{d}_s)$ that intersect e. Then $I_e = \left| \left\lfloor \frac{\mathbf{v} \cdot (\mathbf{d}_s)^\perp}{\delta} \right\rfloor - \left\lfloor \frac{\mathbf{u} \cdot (\mathbf{d}_s)^\perp}{\delta} \right\rfloor \right|$, where $(\mathbf{d}_s)^\perp$ is the direction orthogonal to $\mathbf{d}_s$ and to the left of $\mathbf{d}_s$.

Hence, we can implement this step, by computing $H(\mathbf{d}_s)$ as $(1/2) \sum_e I_e$. This takes $O(n)$ time.

Step 4: Let k be the index such that $\mathbf{d}^{k-1} \prec \mathbf{d}_s \prec \mathbf{d}^k$. Walk along the elements of $\mathbf{D}$ in the order $\mathbf{d}^k, \mathbf{d}^{k+1}, \ldots, \mathbf{d}^{m-1}, \mathbf{d}^0, \ldots, \mathbf{d}^{k-1}$. At each direction $\mathbf{d}^i$, we first compute $H(\mathbf{d}^i)$ from $H(\mathbf{d})$ for $\mathbf{d}^{i-1} \prec \mathbf{d} \prec \mathbf{d}^i$, and then compute $H(\mathbf{d})$ from $H(\mathbf{d}^i)$ for $\mathbf{d}^i \prec \mathbf{d} \prec \mathbf{d}^{i+1}$.

We give some details about this step in Section 4.1. For each direction $\mathbf{d}^i \in \mathbf{D}$, we spend $O(1)$ time to update $H(\mathbf{d})$, so the overall time for Step 4 is $O(|\mathbf{D}|)$.

Step 5: Report the minimum value of $H(\mathbf{d})$ found in Step 4, together with the corresponding optimal hatching direction(s) $\mathbf{d}$.

Theorem 2. *Given a simple polygon $\mathcal{P}$, possibly with holes, having n vertices, Problem 1 can be solved in $O(Cn \log(Cn))$ time, where $C = 1 + \max_v \|v\|/\delta$.*

4.1 Step 4

Let $\mathbf{d}_0$ be any direction of $\mathbf{D}$. We analyze how $H(\mathbf{d})$ changes, if $\mathbf{d}$ rotates in clockwise order, and "passes" through $\mathbf{d}_0$. We denote by $\mathbf{d}_{-\epsilon}$ (resp. $\mathbf{d}_{\epsilon}$) the direction obtained by rotating $\mathbf{d}_0$ by an infinitesimally small angle in counterclockwise (resp. clockwise) direction. Hence, $\mathbf{d}_{-\epsilon}$ (resp. $\mathbf{d}_{\epsilon}$) is the direction $\mathbf{d}$ immediately before it reaches (resp. immediately after it leaves) $\mathbf{d}_0$.

Let v be any vertex of $\mathcal{P}$ that corresponds to $\mathbf{d}_0$, i.e., $d(v, \ell_0(\mathbf{d}_0))$ is a multiple of δ. Let v_p and v_s be the predecessor and successor vertices of v, respectively. Note that the interior of $\mathcal{P}$ is to the left of the directed edges (v_p, v) and (v, v_s). There are two cases, one of which we describe here.

Assume that the points v, $v + d_0$, and v_p or the points v, $v + d_0$, and v_s are collinear. Hence, we have two adjacent vertices, whose (signed) distances to the line $\ell_0(\mathbf{d}_0)$ are equal to the same multiple of δ. We rename these vertices as u and v, and assume w.l.o.g. that the triple $(u, u + d_0^{\perp}, v)$ forms a right-turn. Let u' be the vertex of $\mathcal{P}$ that is adjacent to u and for which $u' \not= v$. Similarly, let v' be the vertex that is adjacent to v and for which $v' \not= u$.

When $\mathbf{d}$ passes through $\mathbf{d}_0$, there are fifty six cases. We consider one of these cases; for the other cases, we refer to [5]. As in Figure 4, assume that (1) $(0, d_0^{\perp}, u)$ forms a right-turn, (2) $(0, d_0^{\perp}, v)$ forms a right-turn, (3) $(u, u + d_0, u')$ forms a left-turn, (4) $(v, v + d_0, v')$ forms a left-turn, and (5) v is the successor of u. (Recall that we assume that $(u, u + d_0^{\perp}, v)$ forms a right-turn.)

We argue that $H(\mathbf{d}_0) = H(\mathbf{d}_{-\epsilon})$, and $H(\mathbf{d}_{\epsilon}) = H(\mathbf{d}_0) - 1$, as follows: Let j be the integer such that $d(u, \ell_0(\mathbf{d}_0)) = d(v, \ell_0(\mathbf{d}_0)) = j\delta$. For any direction $\mathbf{d}$, let $\ell_j(\mathbf{d})$ be the line having direction $\mathbf{d}$ and whose distance to $\ell_0(\mathbf{d})$ is equal to $j\delta$. (Figure 4.) Consider what happens if $\mathbf{d}$ rotates in clockwise order, and passes through $\mathbf{d}_0$. For direction $\mathbf{d}_{-\epsilon}$, the intersection of line $\ell_j(\mathbf{d}_{-\epsilon})$ with $\mathcal{P}$ contains a line segment L, whose endpoints are in the interiors of the edges (u', u) and (v, v'). For direction $\mathbf{d}_0$, the intersection of line $\ell_j(\mathbf{d}_0)$ with $\mathcal{P}$ contains the edge (u, v). If we rotate the direction from $\mathbf{d}_{-\epsilon}$ to $\mathbf{d}_0$, then L "moves" to the edge (u, v). Hence, we indeed have $H(\mathbf{d}_0) = H(\mathbf{d}_{-\epsilon})$. For direction $\mathbf{d}_{\epsilon}$, edge (u, v) does not contribute any line segment to the intersection of line $\ell_j(\mathbf{d}_{\epsilon})$ with $\mathcal{P}$. Therefore, we have $H(\mathbf{d}_{\epsilon}) = H(\mathbf{d}_0) - 1$.

5 Experimental Results

We implemented the 2-dimensional algorithm of Section 3 in C++, and tested it on slices generated from real-world polyhedral models obtained from Stratasys, Inc., a Minnesota-based LM company. We generated the slices using Stratasys' `QuickSlice` program. Figure 5 (top row) displays some of our results.

We also implemented the idea discussed at the end of Section 1.1 to compute a globally optimal direction for all slices. Figure 5 (bottom row) displays some of our results, as viewed in projection in the positive z-direction. (We used a layer thickness of 0.01 inches.) Additional results for both experiments are in [1].

We remark that the approximation algorithms work on polygons with holes in exactly the same way as they do on polygons without holes. In fact, the

algorithms only need the orientation and lengths of the edges; they do not use any information about the adjacency of the edges.

We also implemented the exact algorithm from Section 4. In a separate set of experiments, reported in detail in [5], we tested the exact and approximation algorithms on several additional test files, using now a Sun Ultra with a 400 MHz CPU and 512 MB of RAM. (We ran the algorithms only on single layers, not all layers.) The approximation algorithm generated at most fourteen percent more hatching segments than the exact algorithm. The running time of the exact algorithm ranged from 38 seconds (on a 32-vertex polygon) to 2485 seconds (890 vertices); the approximation algorithm never took more than 1 second.

6 Analysis of the Approximation Algorithm

Our experimental results suggest that the approximation algorithm does well in practice. To further understand its behavior, we also analysed it theoretically.

Let $\delta > 0$ be the width of the tool-tip and n the number of vertices in the polygon $\mathcal{P}$. For any direction $\mathbf{d}$, let $Proj(\mathbf{d}^{\perp})$ be the length of the projection of the edges of $\mathcal{P}$ perpendicular to $\mathbf{d}$, and let $Cut(\mathbf{d})$ be the number of times the boundary of $\mathcal{P}$ is cut when hatched in direction $\mathbf{d}$. Let $\mathbf{d}_p$ and $\mathbf{d}_c$ be the directions minimizing $Proj(\mathbf{d}^{\perp})$ and $Cut(\mathbf{d})$, respectively; $\mathbf{d}_p$ is the direction computed by the approximation algorithm.

In [1], we prove that $Cut(\mathbf{d}_p) - Cut(\mathbf{d}_c) < 3n + (Proj(\mathbf{d}_p^{\perp}) - Proj(\mathbf{d}_c^{\perp}))/\delta$. Since $Proj(\mathbf{d}_p^{\perp}) - Proj(\mathbf{d}_c^{\perp}) \leq 0$, we have that $Cut(\mathbf{d}_p) - Cut(\mathbf{d}_c) < 3n$, or $Cut(\mathbf{d}_p)/Cut(\mathbf{d}_c) < 1 + 3n/Cut(\mathbf{d}_c)$.

If the number of cuts is too small, features will be lost in the model. It is reasonable to assume that $Cut(\mathbf{d}_c) \geq kn$, where $k \geq 1$. This is true if, e.g., many edges of the polygon are cut at least k times. We then have $Cut(\mathbf{d}_p)/Cut(\mathbf{d}_c) < 1 + 3/k$. Furthermore, if in directions $\mathbf{d}_p$ and $\mathbf{d}_c$, each edge is cut in its interior only, then $Cut(\mathbf{d}_c)$ is twice the minimum number of hatching segments and $Cut(\mathbf{d}_p)$ is twice the number of the hatching segments generated by the approximation algorithm. This yields an approximation ratio of $1 + 3/k$.

7 Other Applications

Our methods can solve several related problems efficiently (see [1]):

To improve part strength it is desirable to hatch each slice along two non-parallel directions [3]. This yields the following problem: Given a simple n-vertex polygon $\mathcal{P}$, possibly with holes, and a fixed angle θ, $0 < \theta \leq 90°$, find a pair of directions $(\mathbf{d}, \mathbf{d}')$ that make an angle θ with each other such that the total number of hatching segments for $\mathcal{P}$ in these two directions is minimized. This problem can be converted to a form where the algorithm of Section 2 or Section 3 can be applied, and can be solved in $O(n \log n)$ time and $O(n)$ space.

Suppose that we wish to protect certain functionally critical edges of the slice from being hit too often during hatching. We can assign weights to edges in

proportion to their importance. This leads to a weighted version of Problem 2, which we can solve in $O(n \log n)$ time and $O(n)$ space.

When a polygonal slice is built via LM, certain edges will have a stair-step appearance due to the discretization introduced by the tool-tip width (similar to anti-aliasing in computer graphics). We quantify the error in terms of the total height of the stair-steps on all edges and show how our methods can be used to minimize the total error, again in $O(n \log n)$ time and $O(n)$ space.

We generalize Problem 2 to vectors in $k > 2$ dimensions and present two algorithms: one runs in $O(n^{k-1} \log n)$ time and $O(n)$ space, and the other in $O(n^{k-1})$ time and space. We also present experimental results for $k = 3$, using as input the facet normals of our models.

References

1. M. Hon, R. Janardan, J. Schwerdt, and M. Smid. Minimizing the total projection of a set of vectors, with applications to Layered Manufacturing. Manuscript, January 2001. `http://www.cs.umn.edu/~janardan/min-proj.ps`.
2. M. E. Houle and G. T. Toussaint. Computing the width of a set. *IEEE Trans. Pattern Anal. Mach. Intell.*, PAMI-10(5):761 765, 1988.
3. P. Jacobs. *Rapid Prototyping & Manufacturing: Fundamentals of Stereolithography.* McGraw-Hill, 1992.
4. S. E. Sarma. The crossing function and its application to zig-zag tool paths. *Comput. Aided Design*, 31:881 890, 1999.
5. J. Schwerdt, M. Smid, M. Hon, and R. Janardan. Computing an optimal hatching direction in Layered Manufacturing. Manuscript, January 2001. `http://isgwww.cs.uni-magdeburg.de/~michiel/hatching.ps.gz`.

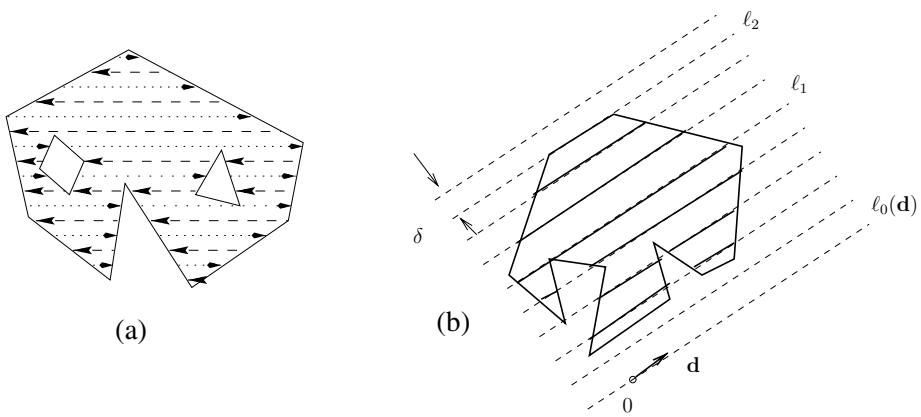


Fig. 1. (a) Hatching a polygonal slice. (b) Formal de nition for hatching problem. Here $H(\mathbf{d}) = 10$. Note that lines ℓ_1 and ℓ_2 each contribute one segment.

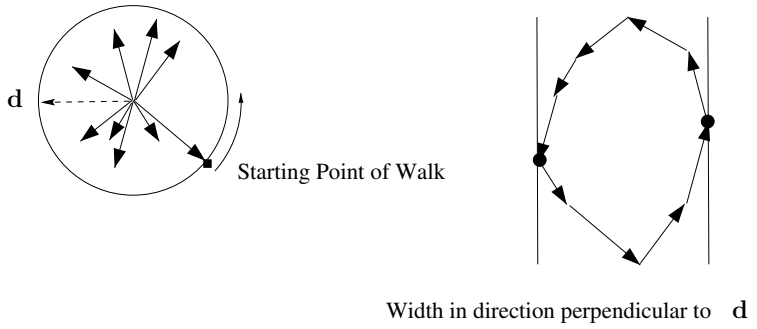


Fig. 2. A set of vectors and the resulting convex polygon. The sum of the absolute values of the dot products of the vectors w.r.t. direction **d** is twice the width of the convex polygon in the direction perpendicular to **d**.

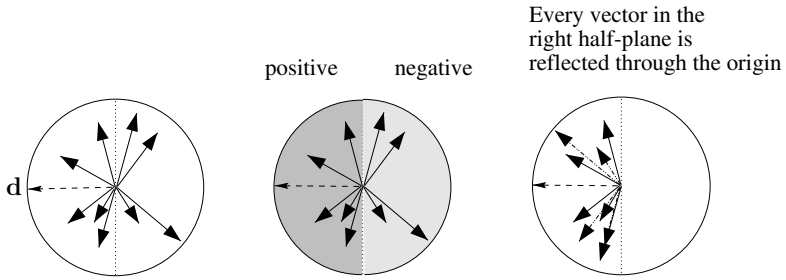


Fig. 3. As an initial step, we pick an arbitrary candidate direction **d** and make sure every vector falls in its positive half-plane. In this figure, the candidate direction is the negative x direction.

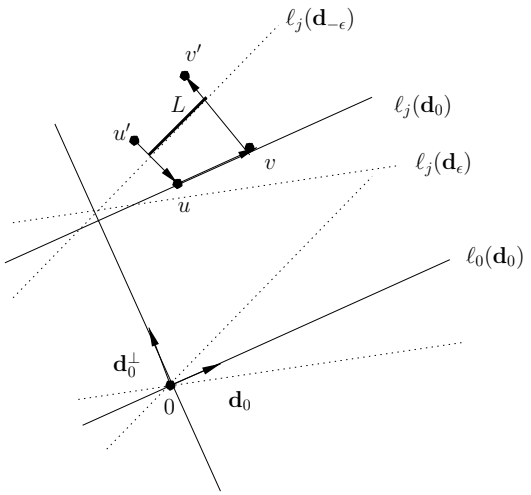


Fig. 4. Illustrating Step 4 in Section 4.1.

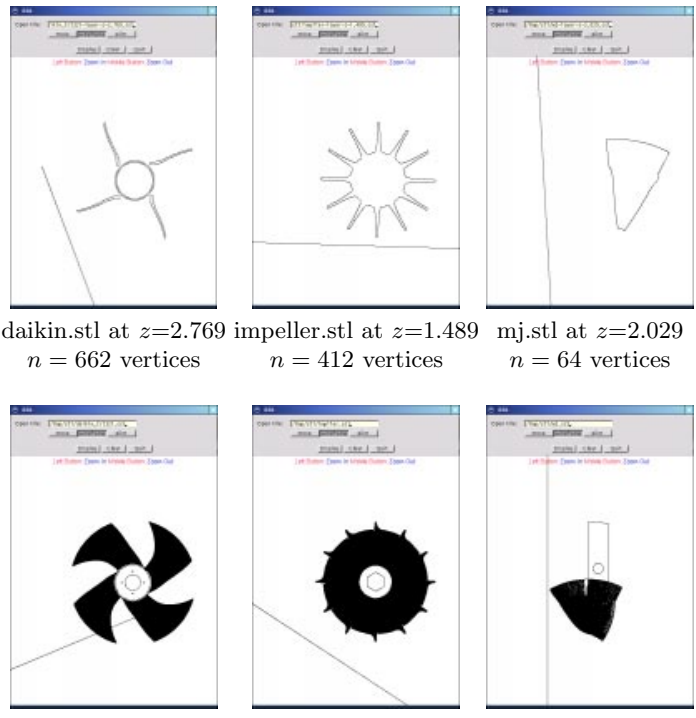


Fig. 5. Screen shots of the program running on a single layer (top row) and all layers (bottom row) of different models. (The z value in the top row shows the height of the layer above the platform.) The long lines inside each window is the resulting hatching direction, which minimizes the sum of the lengths of the projections of the edges onto a perpendicular line. For each model, the running time for a single layer was less than 0.01 seconds and for all layers was less than 2 seconds, on a Sun UltraSparcIIi workstation with a 440 MHz CPU and 256 MB of RAM.

Discrete Local Fairing of B-Spline Surfaces*

Seok-Yong Hong, Chung-Seong Hong, Hyun-Chan Lee, and Koohyun Park

Department of Information and Industrial Engineering,
Hong-Ik University, Sangsu-dong 72-1, Mapo-gu,
Seoul, Republic of Korea (121-791)
sy_hong@orgio.net, pulnaeum@shinbiro.com,
hclee@wow.hongik.ac.kr, khpark@wow.hongik.ac.kr

Abstract. Many surfaces can be modeled by interpolating data points digitized from existing products. But the digitized data points could have measuring errors. To adjust the points, fairing is performed. We present an automatic local fairing algorithm using nonlinear programming. For the objective function of the algorithm, we derive discrete fairness metrics. The metrics are consisted of discrete principal curvatures. The discrete principal curvatures are calculated with the given data points.

1 Introduction

Reverse engineering is popular in product design. In reverse engineering, surfaces can be modeled by interpolating data points digitized from existing products. Designers can model new products by modifying the surfaces. But the digitized points could have noises. If the surfaces are constructed with such points, they have unwanted shapes. Fairing is necessary to adjust the points with noises.

Many existing fairing algorithms bring about good fairing results. However, excessive fairing of surface could have a problem. The problem is that the pattern of local shapes of an original surface is not preserved after fairing [3,5]. Thus, we present a new fairing algorithm, which performs iterative local fairing of data points of B-spline surfaces. As a result, it brings about fair surfaces and preserves the pattern of local shapes of an original surface. Because it performs fairing of data points, we adopted new discrete fairness metrics. The fairness metrics contain discrete principal curvatures. We derived the discrete principal curvatures from the data points.

2 Discrete Fairness Metrics

Fairness criteria are necessary to determine whether a surface is fair or unfair. The presented algorithm can use various fairness criteria such as flattening, rounding, and rolling. Designers can choose a fairness criterion suitable to design intents. Once a fairness criterion is selected, fairness of a surface must be measured numerically. The numerical measure of fairness is called fairness metric.

* This research was supported by Brain Korea 21 grant.

V.N. Alexandrov et al. (Eds.): ICCS 2001, LNCS 2073, pp. 693–697, 2001.
© Springer-Verlag Berlin Heidelberg 2001

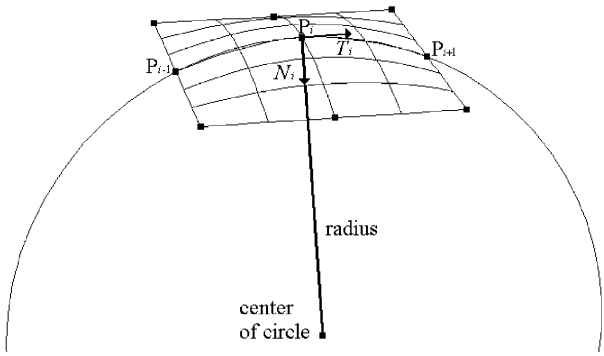


Fig. 1. Calculating discrete nomal curvatures with data points

We adopted the concept of derived surface to calculate discrete fairness metrics used in fairing data points [5]. A derived surface is consisted of geometric invariants such as curvature, radius of curvature, torsion, unit normal vector, unit tangent vector, and unit binormal vector. Once the derived surface for a fairness criterion is determined, we can derive a fairness metric by calculating the area of the derived surface. The fairness metric is used as the objective function of an optimization problem for fairing. If we minimize the objective function, the surface under consideration becomes fair. One of the fairness metrics we used is a rolling metric. If the rolling metric is minimized, an original surface tends to be made more cylindrical or conical. The rolling metric is shown in the equation (1). In the equation, W is defined as $K + H^2$. K, H, k_1, and k_2 denote Gaussian curvature, mean curvature, and two principal curvatures, respectively. s and t are the parameters of the surface to be faired.

$$\int \int |W| \left[\left(k_1 \frac{\partial W}{\partial t} \right)^2 + \left(k_2 \frac{\partial W}{\partial s} \right)^2 + W^2 K^2 \right]^{1/2} dsdt \tag{1}$$

To adopt the concept of the fairness metrics stated above in fairing data points, discrete fairness metrics must be derived. A discrete fairness metric is calculated with data points and it is consisted of discrete principal curvatures. To calculate the discrete principal curvatures at a data point, discrete normal curvatures at the data point must be derived. If data points are positioned in rectangular manner, a data point is surrounded by eight neighboring data points as shown in Fig. 1. Thus, discrete normal curvatures at the data point can be calculated in four directions, because they are calculated using the circles passing through three consecutive points containing the data point. The four discrete normal curvatures can be calculated using the original concept of normal curvature. The original normal curvature κ_n at a point on a surface can be calculated as follows [1]:

$$\kappa_n = \kappa N \cdot n \tag{2}$$

The four directional discrete normal curvatures at a data point can be calculated as follows:

First, discrete curvature at the data point is substituted for the curvature κ_n in the equation (2). As shown in Fig. 1, with the given three consecutive points, P_{i-1}, P_i, and P_{i+1}, discrete curvature κ_i at the point P_i is calculated as the inverse of the radius of the circle passing through the three points. In addition, its first derivative can be calculated with discrete curvatures and the distances between the given points [2].

Second, the discrete main normal vector N_i at the same data point is substituted for the main normal vector N of a curve on a surface. It can be calculated as the unit vector, of which direction is from the data point to the center of the circle passing through the three points used in calculating the discrete curvature at the data point.

Third, to approximate the unit normal vector n of a surface, we calculate four directional unit tangent vectors with the data point and eight neighboring data points. In Fig. 1, T_i denotes one of the four unit tangent vectors. Each of the four vectors is tangent to the corresponding circle passing through the three consecutive points used for the discrete curvature. Then, we calculate two cross product vectors for two pairs of unit tangent vectors. One pair of the unit tangent vectors are of horizontal and vertical directions. The other pair of the vectors are the unit tangent vectors of diagonal directions. Finally, we can approximate the unit normal vector n by calculating the average of the two cross product vectors.

Therefore, we can calculate four directional discrete normal curvatures at a data point using the derived discrete curvatures and normal vectors. Then we select a minimum discrete normal curvature and the other discrete normal curvature of orthogonal direction. The two curvatures are the discrete principal curvatures [4]. The discrete principal curvatures and the first derivatives of the discrete principal curvatures are used in calculating discrete fairness metrics.

3 Discrete Fairing Algorithm

The proposed discrete fairing algorithm performs iterative local fairing. For local fairing, a local fairness function is chosen as objective function. It evaluates fairness at a data point. Then, global fairness function is evaluated by accumulating all local fairness function values. It evaluates fairness for whole data points. The procedures of the algorithm are as follows:

Step 1. A data point set is given as input data.

Step 2. For a fairness metric, the value of the local fairness function for each data point and the value of the global fairness function are calculated.

Step 3. The point of which value of the local fairness function is the largest is selected.

Step 4. An optimization problem is formulated for improving fairness at the selected data point as follows. First, the local fairness function for the selected

data point is used as the objective function of the optimization problem. Second, the free variables of the problem are the coordinates of the selected data point. Third, a constraint is set from the distance measure between the original data point and the modified data point. Then, new data point is calculated through the optimization process.

Step 5. If new value of the global fairness function is reduced, take the new point as the modified data point and go to step 2. If it is not reduced, the point of which value of the local fairness function is the next largest is selected and go to step 4. However, if no more point exists to select, go to step 6.

Step 6. A new B-spline surface is constructed by interpolating the modified data point set.

4 Experimental Results

We tested the proposed discrete fairing algorithm with an example data point set. Fig. 2 shows the results of fairing with a rolling metric. The figure contains an original surface, the surface faired with the discrete fairing algorithm, and the surface faired with an analytic fairing algorithm. The original surface is constructed by interpolating 63 data points. The analytic fairing algorithm is one of the existing fairing algorithms. It performs global fairing and uses analytic fairness metrics derived from surface geometry [5]. As the right view shows, the discrete fairing algorithm fairs the original surface and preserves the local shapes of the original surface, while the analytic fairing algorithm removes all of the local shapes. Fig. 3 shows the mean curvature graphs for the fairing results and explains the fairing results well. Because the analytic fairing algorithm performs excessive fairing, the mean curvature graph of the surface faired with it is almost flat. However, the mean curvature graph of the surface faired with the discrete fairing algorithm is smoother than the original mean curvature graph. In addition, the pattern of local shapes of the original mean curvature graph is preserved. The right view shows these results clearly. The reason of the shape preservation comes from the fact that the proposed discrete local fairing algorithm improves fairness at fewer points, which cause only local irregularities.

5 Conclusions

We proposed a discrete fairing algorithm. The algorithm performs local and discrete fairing of data points of B-spline surfaces. It fairs the data points and preserves the local shapes of an original surface better than existing global fairing algorithms. This is due to its local and discrete fairing. In addition, the algorithm has less computation time, because it has one free point per a fairing iteration and uses discrete differential geometry.

Therefore, when designers want to preserve the pattern of local shapes of an initial surface after fairing, the proposed algorithm can be adopted and used. There are two future works to be continued. One is to develop another fairness

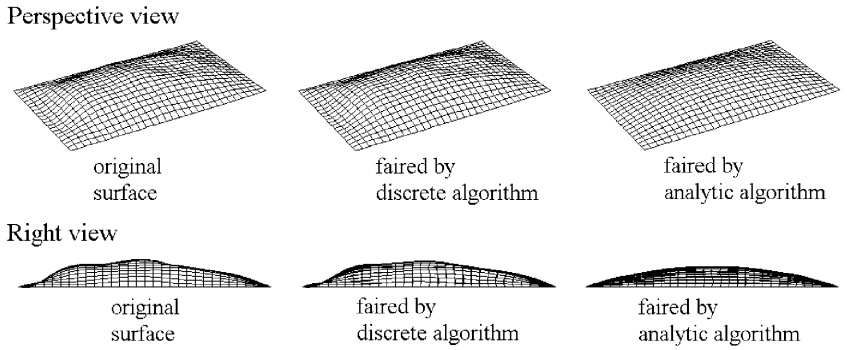


Fig. 2. Fairing results with rolling metric

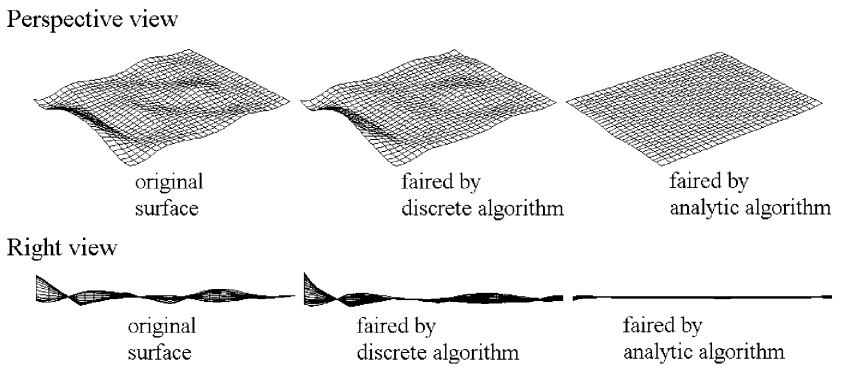


Fig. 3. Mean curvature graphs for the fairing results with rolling metric

metrics and the other is to construct a new surface fairing algorithm, which does not adopt optimizations but uses analytic equations to improve fairness.

References

1. Choi, B. K.: Surface Modeling for CAD/CAM. Elsevier, Amsterdam Oxford New York Tokyo (1991) 25-29
2. Eck, M. and Jaspert, R.: Automatic Fairing of Point Sets. Designing Fair Curves and Surfaces. Society for Industrial and Applied Mathematics, Philadelphia (1994) 45-60
3. Lott, N. J. and Pullin, D. I.: Method for Fairing B-spline Surfaces. Computer-Aided Design. 10 (1988) 597-604
4. O'Neill, B.: Elementary Differential Geometry. Academic Press (1966) 199-202
5. Rando, T. and Roulier, J.A.: Measures of Fairness for Curves and Surfaces. Designing Fair Curves and Surfaces. Society for Industrial and Applied Mathematics, Philadelphia (1994) 75-122

Computational Methods for Geometric Processing. Applications to Industry

A. Iglesias, A. Gálvez, and J. Puig-Pey

Department of Applied Mathematics and Computational Sciences, University of Cantabria, Avda. de los Castros, s/n, E-39005, Santander, Spain
iglesias@unican.es

Abstract. This paper offers a unifying survey of some of the most relevant computational issues appearing in geometric processing (such as blending, trimming, intersection of curves and surfaces, offset curves and surfaces, NC milling machines and implicitization). Applications of these topics to industrial environments are also described.

1 Introduction

Geometric processing is defined as the calculation of geometric properties of already constructed curves, surfaces and solids [5]. In its most comprehensive meaning, this term includes all the algorithms that are applied to already existing geometric entities [16]. As pointed out in [5], since geometric processing is intrinsically *hard* there is neither a unified approach nor "key developments" such as the Bézier technique [60] for design. On the contrary, the literature on geometric processing is much more disperse among different sources. The aim of the present paper is precisely to offer a unifying survey of some of the most relevant computational issues appearing in geometric processing as well as a description of their practical applications in industry. Obviously, this task is too wide to be considered in all its generality, and some interesting topics in geometric processing, such as curvature analysis, contouring, curve fairing, etc. have been omitted. We restrict ourselves to blending (Section 2.1), trimmed surfaces (Section 2.2), curve and surface intersection (Section 2.3), offset curves and surfaces (Section 2.4), NC milling technology (Section 2.5) and implicitization (Section 2.6).

2 Some Geometric Processing Topics

2.1 Blend Surfaces

We use the term *blending* to mean the construction of connecting curves and surfaces and the rounding off of sharp corners or edges. Thus, we talk about *superficial blending* to indicate that no explicit mathematical formula is available. It appears in the production process [87,88], in procedures such as round off a corner or edge with radius r. The blend described by additional surfaces connecting smoothly some given surfaces is usually referred to as *surface blending*,

V.N. Alexandrov et al. (Eds.): ICCS 2001, LNCS 2073, pp. 698–707, 2001.
© Springer-Verlag Berlin Heidelberg 2001

while the *volumetric blending* is used to mean the combination of objects in a solid modeling system (see [34], Chapter 14).

The most interesting blend for our purposes is that in parametric form. To this aim, a number of methods are described, from *interactive methods* [4,56] to *automatic methods* based on calculation of intersections of offset surfaces to the two given surfaces [46,56]. Blending of tensor product B-spline or Bézier surfaces (see [18,20,34] for a definition) are analyzed, for example, in [4,12,24,45]. See also [86] for blending algebraic patches and [28,66] for implicit surfaces.

2.2 Trimmed Surfaces

Trimmed surfaces have a fundamental role in CAD. Most complex objects are generated by some sort of trimming/scissoring process, i.e. unwanted parts of the rectangular patch are trimmed away (see Fig. 1). Trimmed patches are also the result of Boolean operations on solid objects bounded by NURBS surfaces (see [19,61,68] for a definition). In the computer-aided design pipeline, the trimmed patch undergoes a number of processes such as rendering for visualization, cutter path generation, area computation or rapid prototyping, also known as *solid hard copy* [79]. For visualization, trimmed surfaces are rendered in two stages [67,77]: the surface is divided into a number of planar tesselants (triangles or other polygons), which are rendered using standard methods for planar polygons. Other algorithms for tessellation of trimmed NURBS surfaces can be found in [63] (and references 6-19 therein).

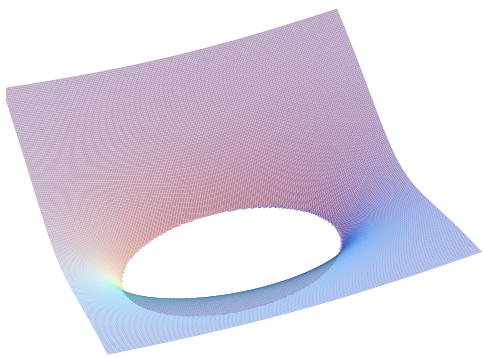


Fig. 1. Example of a trimmed NURBS surface

2.3 Intersection of Curves and Surfaces

In many applications, computation of the intersections of curves and surfaces is required. Among them, we quote smooth blending of curves and surfaces

(Section 2.1), the construction of contour maps to visualize surfaces, Boolean operations on solid bodies and determination of self-intersections in offset curves and surfaces (Section 2.4).

There exists a significant body of literature on the calculation of intersections of two parametric surfaces [1,6,18,23,30,76] (see also [17] for a more exhaustive bibliography). Recent developments include the possibility of handling intersection singularities [10,49]. Intersections of offsets (see Section 2.4) of parametric surfaces are analyzed in [85]. This problem is often of great interest: for instance, a blend surface (see Section 2.1) of two surfaces can be constructed by moving the center of a sphere of given radius along the intersection curve of two surfaces that are offset from the base surfaces by the radius of the sphere.

However, there has been no known algorithm that can compute the intersection curve of two arbitrary rational surfaces accurately, robustly and efficiently [34]. In addition, it is known that two surface patches intersect in a curve whose degree is much higher than the parametric degree of the two patches. Thus, two bicubic patches intersect in a curve of degree 324!!! Fortunately, the situation is better when we restrict the domain of input surfaces to simple surfaces (planes, quadrics and tori, i.e. the so-called *CSG primitives*) [43,53,78]. These surfaces are important in conventional solid modeling systems for industry, since they can represent a large number of mechanical parts of a car, ship, plane, etc.

As noticed in the previous paragraph, algorithms for intersections strongly depend on the general form of the curves and surfaces we are dealing with. If both objects are given in implicit form, such an intersection is found by solving a system of nonlinear equations. This can be achieved through numerical methods [23], differential geometry [3] or a combination of geometric and analytic methods [54]. If the objects are described as free-form curves and surfaces [18,20,23,34, 61,68], methods can be grouped into several categories: *algebraic methods*, based on implicitization (Section 2.6), *subdivision methods*, which divide the objects to be intersected into many pieces and check for intersections of the pieces [6,9,13, 26,27,42,47,91], *discretization methods*, which reduce the degrees of freedom by discretizing the surface representation in several ways, such as contouring [14,58, 81] or parameter discretization [6,35], *hybrid methods*, which combine subdivision and numerical methods [82,90], etc.

2.4 Offset Curves and Surfaces

Offsetting is a geometric operation which expands a given object into a similar object to a certain extent. In general, we deal with offset curves and surfaces, which are also curves and surfaces at a constant distance d from a given initial curve or surface. Several methods for the computation of the offsets for curves are compared in [15]. As pointed out in [59], offsetting general surfaces is more complicated, and an offset surface is often approximated [21], although this approximation becomes inaccurate near its selfintersecting area [2,59]. Another approach for computing offsets of NURBS curves and surfaces is given in [62].

Offsetting has various important applications [69]. For example, if the inner surface of a piece is taken as the reference surface, the outer surface can be

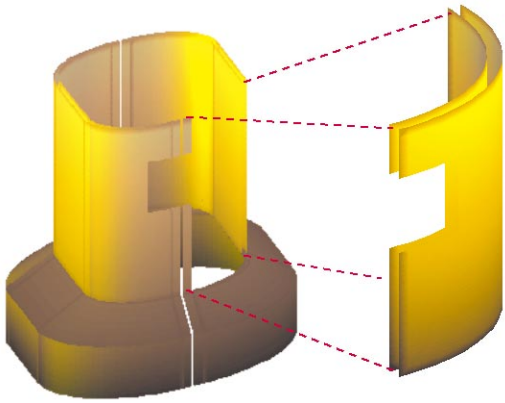


Fig. 2. Application of the offset operation: the outer surface of the piece is the offset of the inner trimmed NURBS surface

mathematically described by an offset surface corresponding to a distance equal to the thickness of the material (see Fig. 2). Offsets also appear in cutter-path generation for numerical control machine tools: pieces of a surface can be cut, milled or polished using a laser-controlled device to follow the offset. In the case of curves, they can be seen as the envelope corresponding to moving the center of a circle of radius d along the initial curve. This allows to define both the inside and outside offset curves, with applications in milling. Finally, they are fundamental tools (among others) in the constant-radius rounding and filleting of solids or in tolerance analysis, for definition of tolerance zones, etc.

We should note, however, that offset curves and surfaces lead to several practical problems. Depending on the shape of the initial curve, its offset can come closer than d to the curve, thus causing problems with *collisions*, for instance, when steering a tool. These collision problems also arise in other applications, as path-planning for robot motions, a key problem in the current industry. To avoid this, we need to remove certain segments of the curve which start and end at self-intersections [29,70]. Special methods for the case of interior offsets (as used in milling holes or *pockets*) can be found in [29] and [57]. In the case of surfaces, the scenario is, by large, much more complicated: singularities at a point can arise when the distance d of the smallest value of the principal curvature is attained at the point. In addition, these singularities can be of many different types: cusps, sharp edges or self-intersections [21]. Finally, the set of rational curves and surfaces is not closed under offsetting [18]. Therefore, considerable attention has been paid to identify the curves and surfaces which admit rational offsets [22,59,64]. The case of polynomial and rational curves with rational offsets is analyzed in [48]. We also recommend [50] for a more recent overview of offset curves and surfaces.

Other recent developments are *geodesic offsets* [55] and *general offsets*, first introduced in [7] and extended in [65]. Both kinds of offsets exhibit applications in manufacture. For example, geodesic offset curves are used to generate tool paths on a part for zig-zag finishing using 3-axis machining (see Section 2.5) with ball-end cutter so that the scallop-height (the cusp height of the material removed by the cutter) will become constant. This leads to a significant reduction in size of the cutter location data and hence in the machining time. On the other hand, not only ball-end but also cylindrical and toroidal cutters are used in 3-axis NC machining. When the center of the ball-end cutter moves along the offset surface, the reference point on the cylindrical and toroidal cutters move along the general offset.

2.5 NC Milling

Numerical controlled (NC) milling technology is a process where a rotating cutter is sequentially moved along prescribed tool paths in order to manufacture a free-form surface from raw stock. NC milling is an essential tool for manufacturing free-form surfaces. For example, dies and injection molds for automobile parts are manufactured by using milling machines, which can be classified as a function of the number of axis in two (used to cut holes [29,57]), two-and-one-half, three, four and five axis (to mill free-form surfaces) (see [34], Chapter 16). These tasks have given rise to a number of different problems [44], such as those related to the determination of the milling coordinates and axis relative to the desired surface depending on the type of milling, transformation of control curves to machine coordinates, displacement of the tool along special surface curves or collision checking, etc. In general, these problems can be summarized as the determination of which parts of the surface are effected as the milling tool moves.

At first sight, two different approaches for the simulation of the process can be considered [25]: the exact, analytical approach [41,80] (which is computationally expensive) and the approximation approach. The cost of the simulation for the first approach (when using Constructive Solid Geometry) is reported to be $O(n^4)$ (n being the number of tool movements) by $O(n)$ for the approximation approach [38]. Since a complex NC program might consist of ten thousand movements, the first approach is computationally unapproachable and only approximate techniques are applied [32,36,37,38,72].

2.6 Implicitization

In the last years, implicit representations are being used more frequently in CAGD, allowing a better treatment of several problems. As one example, the point classification problem is easily solved with the implicit representation: it consists of a simple evaluation of the implicit functions. This is useful in many applications, as solid modeling for mechanical parts, for example, where points must be defined inside or outside the boundaries of an object, or for calculating intersections of free-form curves and surfaces (see Section 2.3). Through implicit representation, the problem is reduced to a trivial sign test. Other advantages are

that the class of implicit surfaces is closed under such operations as offsetting, blending and bisecting. In other words, the offset (see Section 2.4) of an algebraic curve (surface) is again an algebraic curve (surface) and so on. In addition, the intersection (see Section 2.3) of two algebraic surfaces is an algebraic curve. Furthermore, the implicit representation offers surfaces of desired smoothness with the lowest possible degree. Finally, the implicit representation is more general than the rational parametric one [30]. All these advantages explain why the implicit equation of a geometric object is of importance in practical problems.

Implicitization is the process of determining the implicit equation of a parametrically defined curve or surface. One remarkable fact is that this parametric-implicit conversion is *always* possible [11,75]. Therefore, for any parametric curve or surface there exists an implicit polynomial equation defining exactly the same curve or surface. The corresponding algorithm for curves is given in [73] and [74]. In addition, a parametric curve of degree n has an implicit equation of also degree n. Further, the coefficients of this implicit equation are obtained from those of the parametric form by using only multiplication, addition and subtraction, so conversion can be performed through symbolic computation, with no numerical error introduced. Implicitization algorithms also exist for surfaces [51,73, 74]. However, a triangular parametric surface patch of degree n has an implicit equation of degree n^2. Similarly, a tensor product parametric patch of degree (m, n) has an implicit equation of degree $2mn$. For example, a bicubic patch has an implicit equation of degree 18 with 1330 terms!!!

In general, the implicitization algorithms are based on *resultants*, a classical technique [71], Gröbner bases techniques [8] and on the Wu-Ritt method [89]. Resultants provide a set of techniques [39] for eliminating variables from systems of nonlinear equations. However, the derived implicit equation may have extraneous factors: for example, surfaces can exhibit additional sheets. On the other hand, symbolic computation required to obtain the implicit expression exceeds the resources in space and time, although parallel computation might, at least partially, solve this problem. On the other hand, given an initial set of two or three polynomials defining the parametric curve or surface as a basis for an ideal [30], the Gröbner basis will be such that it contains the implicit form of the curve or surface. In the rational case, additional polynomials are needed to account for the possibility of base points [40]. Finally, the Wu-Ritt method consists of transforming the initial set into a triangular system of polynomials. This transformation involves rewriting the polynomials using pseudo-division and adding the remainders to the set. The reader is referred to [39] and [89] for more details. With respect to implementation, hybrid symbolic/numerical methods have been proposed in [52]. Also, in [31] atractive speed-ups for Gröbner based implicitization using numerical and algebraic techniques have been obtained.

Finally, we remark that implicitization can be seen as a particular case of conversion between different curve or surface forms (see, for example, [83,84]). See also [33] (and references therein) for a survey on approximate conversion between Bézier and B-spline surfaces, which are also applied to offsets.

Acknowledgements

The authors would like to acknowledge the CICYT of the Spanish Ministry of Education (project TAP98-0640) and the European Fund FEDER (Contract 1FD97-0409) for partial support of this work. They also thank the referees for their careful reading of the initial version of the manuscript and their helpful suggestions which allowed a substantial improvement of the paper.

References

1. K. Abdel-Malek and H.J. Yeh: On the determination of starting points for parametric surface intersections. CAD **29** (1997) 21-35
2. S. Aomura and T. Uehara: Self-intersection of an offset surface. CAD **22** (1990) 417-422
3. C. Asteasu: Intersection of arbitrary surfaces. CAD **20** (1988) 533-538
4. L. Bardis and N.M. Patrikalakis: Blending rational B-spline surfaces. Eurographics'89 (1989) 453-462
5. R.E. Barnhill: Geometry Processing for Design and Manufacturing, SIAM, Philadelphia, PA (1992)
6. R.E. Barnhill and S.N. Kersey: A marching method for parametric surface/surface intersection. CAGD **7** (1990) 257-280
7. E.L. Brechner: General tool offset curves and surfaces. In: R.E. Barnhill (ed.): Geometry Processing for Design and Manufacturing, SIAM (1992) 101-121
8. B. Buchberger: Gröbner bases: an algorithmic method in polynomial ideal theory. In: N.K. Rose (ed.): Multidimensional Systems theory, Reidel Publishing Co. (1985) 184-232
9. W.R. Carlson: An algorithm and data structure for 3D object synthesis using surface patch intersections. Computer Graphics **16** (1982) 255-263
10. E.W. Chionh and R.N. Goldman: Using multivariate resultants to find the implicit equation of a rational surface. The Visual Computer **8** (1992) 171-180
11. K.P. Cheng: Using plane vector fields to obtain all the intersection curves of two general surfaces. In: W. Strasser and H.P. Seidel (ed.): Theory and Practice in Geometric Modeling, Springer, New York (1989) 187-204
12. B.K. Choi and S.Y. Ju: Constant-radius blending in surface modeling. CAD **21** (1989) 213-220
13. E. Cohen, T. Lyche and R.F. Riesenfeld: Discrete B-splines and subdivision techniques in CAGD and computer graphics. Computer Graphics and Image Processing **14** (1980) 87-111
14. D.P. Dobkin, S.V.F. Levy, W.P. Thuston and A.R. Wilks: Contour tracking by piecewise linear approximations. ACM Trans. on Graph. **9** (1990) 389-423
15. G. Elber, I. Lee and M.S. Kim: Comparing offset curve approximation methods. IEEE Comp. Graph. and Appl. **17**(3) (1997) 62-71
16. G. Farin: Trends in curve and surface design. CAD **21**(5) (1989) 293-296
17. G. Farin: An ISS bibliography. In: R.E. Barnhill (ed.): Geometry Processing for Design and Manufacturing, SIAM (1992) 205-207
18. G. Farin: Curves and Surfaces for Computer Aided Geometric Design, Fourth Edition, Academic Press, San Diego (1996)
19. G. Farin: NURB Curves and Surfaces: from Projective Geometry to Practical Use, Second Edition, AK Peters, Wellesley, MA (1999)

20. G. Farin and D. Hansford: The Essentials of CAGD, AK Peters, Wellesley, MA (2000)
21. R.T. Farouki: The approximation of non-degenerate offset surfaces. CAGD **3** (1986) 15-43
22. R.T. Farouki: Pythegorean-hodograph curves in practical use. In: R.E. Barnhill (ed.): Geometry Processing for Design and Manufacturing, SIAM (1992) 3-33
23. I.D. Faux and M.J. Pratt: Computational Geometry for Design and Manufacture, Ellis Horwood, Chichester (1979)
24. D.J. Filip: Blending parametric surfaces. ACM Trans. on Graph. **8**(3) (1989) 164-173
25. G. Glaeser and E. Gröller: Efficient volume-generation during the simulation of NC-milling. In: H.C. Hege and K. Polthier (ed.): Mathematical Visualization. Algorithms, Applications and Numerics, Springer Verlag, Berlin (1998) 89-106
26. R.N. Goldman: Subdivision algorithms for Bézier triangles. CAD **15** (1983) 159-166
27. J.G. Griffiths: A data structure for the elimination of hidden surfaces by patch subdivision. CAD **7** (1975) 171-178
28. E. Hartmann: Blending of implicit surfaces with functional splines. CAD **22** (1990) 500-506
29. M. Held: On the computational geometry of pocket machining. Lectures Notes in Computer Science, 500, Springer Verlag, Berlin, New York (1991)
30. C.M. Hoffmann: Geometric and Solid Modeling, Morgan Kaufmann, San Mateo, CA (1989)
31. C.M. Hoffmann: Algebraic and numerical techniques for offsets and blends. In: S. Micchelli, M. Gasca and W. Dahmen (ed.): Computations of Curves and Surfaces, Kluwer Academic (1990) 499-528
32. T. van Hook: Real time shaded NC milling display. Computer Graphics **20**(4) (1986) 15-20 (Proc. SIGGRAPH'86)
33. J. Hoschek and F.J. Schneider: Approximate spline conversion for integral and rational Bézier and B-spline surfaces. In: R.E. Barnhill (ed.): Geometry Processing for Design and Manufaturing, SIAM (1992) 45-86
34. J. Hoschek and D. Lasser: Fundamentals of Computer Aided Geometric Design, A.K. Peters, Wellesley, MA (1993)
35. E.G. Houghton, R.F. Emnett, J.D. Factor and C.L. Sabharwal: Implementation of a divide-and-conquer method for intersection of parametric surfaces. CAGD **2** (1985) 173-183
36. Y. Huang and J.H. Oliver: NC milling error assessment and tool path correction. Computer Graphics Proceedings (1994) 287-294 (Proc. SIGGRAPH'94)
37. K.C. Hui: Solid sweeping in image space-application in NC simulation. The Visual Computer **10** (1994) 306-316
38. R.B. Jerard, S.Z. Hussaini, R.L. Drysdale and B. Schaudt: Approximate methods for simulation and verification on NC machining programs. The Visual Computer **5** (1989) 329-348
39. D. Kapur and Y.N. Lakshman: Elimination methods. In: B. Donald, D. Kapur and J. Mundy (ed.): Symbolic and Numerical Computing for Artificial Intelligence, Academic Press (1992)
40. M. Kalkbrener: Implicitization of rational parametric curves and surfaces. Technical Report, Kepler Universität, Linz, Austria, RISC, Linz (1990)
41. Y. Kawashima, K. Itoh, T. Ishida, S. Nonaka and K. Ejiri: A flexible quantitative method for NC machining verification using a space-division based solid model. The Visual Computer **7** (1991) 149-157

42. T.L. Kay and J.T. Kajiya: Ray tracing complex scenes. Computer Graphics **20** (1986) 269-278
43. K.J. Kim and M.S. Kim: Torus/sphere intersection based on configuration space approach. Graphical Models and Image Processing **60**(1) (1998) 77-92
44. R. Klass and P. Schramm: NC milling of CAD surface data. In: H. Hagen and D. Roller (ed.): Geometric Modeling. Methods and Applications, Springer Verlag, Berlin Heidelberg (1991) 213-226
45. R. Klass and B. Kuhn: Fillet and surface intersections defined by rolling balls. CAGD **9** (1992) 185-193
46. P.A. Koparkar: Designing parametric blends: surface model and geometric correspondence. The Visual Computer **7** (1991) 39-58
47. D. Lasser: Intersection of parametric surfaces in the Bernstein-Bézier representation. CAGD **3** (1986) 186-192
48. W. Lü: Offset-rational parametric plane curves. CAGD **12** (1995) 601-616
49. W. Ma and Y.S. Lee: Detection of loops and singularities of surface intersections. CAD **30** (1998) 1059-1067
50. T. Maekawa: An overview of offset curves and surfaces. CAD **31** (1999) 165-173
51. D. Manocha and J. F. Canny: Algorithm for implicitizing rational parametric surfaces. CAGD **9** (1992) 25-50
52. D. Manocha and J. F. Canny: Implicit representations of rational parametric surfaces. J. of Symbolic Computation **13** (1992) 485-510
53. J. Miller and R.N. Goldman: Geometric algorithms for detecting and calculating all conic sections in the intersection of any two natural quadric surfaces. Graphical Models and Image Processing **57**(1) (1995) 55-66
54. J.C. Owen and A.P. Rockwood: Intersection of general implicit surfaces. In: G.E. Farin (ed.): Geometric Modeling: Algorithms and New Trends, SIAM (1987) 335-345
55. N.M. Patrikalakis and L. Bardis: Offsets of curves on rational B-spline surfaces. Engineering with Computers **5** (1989) 39-46
56. J. Pegna and D.J. Wilde: Spherical and circular blending of functional surfaces. Trans. of ASME, Journal of Offshore Mechanics and Artic Engineering **112** (1990) 134-142
57. H. Persson: NC machining of arbitrarily shaped pockets. CAD **10** (1978) 169-174
58. G. Petrie and T.K.M. Kennie: Terrain modeling in surveying and civil engineering. CAD **19** (1987) 171-187
59. B. Pham: Offset curves and surfaces: a brief survey. CAD **24** (1992) 223-229
60. L. Piegl: Key developments in Computer-Aided Geometric Design, CAD **21**(5) (1989) 262-273
61. L. Piegl and W. Tiller: The NURBS Book, Second Edition, Springer Verlag, Berlin Heidelberg (1997)
62. L. Piegl and W. Tiller: Computing offsets of NURBS curves and surfaces. CAD **31** (1999) 147-156
63. L. Piegl and W. Tiller: Geometry-based triangulation of trimmed NURBS surfaces. CAD **30** (1998) 11-18
64. H. Pottmann: Rational curves and surfaces with rational offsets. CAGD **12** (1995) 175-192
65. H. Pottmann: General offset surfaces. Neural, Parallel and Scientific Computations **5** (1997) 55-80
66. A. Rockwood: The displacement method for implicit blending of surfaces in solid modeling. ACM Trans. on Graph. **8**(4) (1989) 279-297
67. A. Rockwood, K. Heaton and T. Davis: Real-time rendering of trimmed surfaces. Computer Graphics **23** (1989) 107-116 (Proc. SIGGRAPH'89)

68. D.F. Rogers: An Introduction to NURBS: with Historical Perspective, Morgan Kaufmann, San Mateo, CA (2000)
69. J.R. Rossignac and A.A.G. Requicha: Offsetting operations in solid modeling. CAGD **3** (1986) 129-148
70. S.E.O. Saeed, A. de Pennington and J.R. Dodsworth: Offsetting in geometric modeling. CAD **20** (1988) 67-74
71. G. Salmon: Lessons Introductory to the Modern Higher Algebra, G.E. Stechert & Co., New York (1885)
72. T. Saito and T. Takahashi: NC machining with G-buffer method. Computer Graphics **25**(4) (1991) 207-216 (Proc. SIGGRAPH'91)
73. T.W. Sederberg: Implicit and parametric curves and surfaces for computer aided geometric design. Ph.D. thesis, Purdue Univ., West Lafayette, IN (1983) 29-42
74. T.W. Sederberg, D.C. Anderson and R.N. Goldman: Implicit representation of parametric curves and surfaces. Computer Vision, Graphics and Image Processing **28** (1984) 72-74
75. T.W. Sederberg: Algebraic geometry for surface and solid modeling. In: G.E. Farin (ed.): Geometric Modeling: Algorithms and New Trends, SIAM (1987) 29-42
76. T.W. Sederberg and R.J. Meyers: Loop detection in surface patch intersections. CAGD **5** (1988) 161-171
77. M. Shantz and S.L. Chang: Rendering trimmed NURBS with adaptive forward differences. Computer Graphics **22** (1988) 189-198 (Proc. SIGGRAPH'88)
78. C.K. Shene and J. Johnstone: On the lower degree intersections of two natural quadrics. ACM Trans. on Graphics **13**(4) (1994) 400-424
79. X. Sheng and B.E. Hirsch: Triangulation of trimmed surfaces in parametric space. CAD **24**(8) (1992) 437-444
80. A.I. Sourin and A.A. Pasko: Function representation for sweeping by a moving solid. IEEE Trans. on Visualization and Computer Graphics **2**(2) (1996) 11-18
81. D.C. Sutcliffe: Contouring over rectangular and skewed rectangular grids. In: K. Brodlie (ed.): Mathematical Methods in Computer Graphics and Design, Academic Press (1980) 39-62
82. M. Sweeney and R. Bartels: Ray tracing free-form B-spline surfaces. IEEE Comp. Graph. and Appl. **6** (1986) 41-49
83. A.E. Vries-Baayens: Conversion of a Composite Trimmed Bézier Surface into Composite Bézier Surfaces. In: P.J. Laurent, Le Mehaute and L.L.Schumaker (ed.): Curves and Surfaces in Geometric Design, Academic Press, Boston, USA (1991) 485-489
84. A.E. Vries-Baayens and C.H. Seebregts: Exact Conversion of a Composite Trimmed Nonrational Bézier Surface into Composite or Basic Nonrational Bézier Surfaces. In: H. Hagen (ed.): Topics in Surface Modeling, SIAM, Philadelphia, USA (1992) 115-143
85. Y. Wang: Intersections of offsets of parametric surfaces. CAGD **13** (1996) 453-465
86. J. Warren: Blending algebraic surfaces. ACM Trans. on Graph. **8**(4) (1989) 263-278
87. D.B. Welborun: Full three-dimensional CAD/CAM. CAE Journal **13** (1996) 54-60, 189-192
88. J.R. Woodwark: Blends in geometric modeling. In: R.R. Martin (ed.): The Mathematics of Surfaces II, Oxford Univ. Press (1987) 255-297
89. W.T. Wu: Basic principles of mechanical theorem proving in geometries. J. of Systems Sciences and Mathematical Sciences **4** (1986) 207-235
90. C.G. Yan: On speeding up ray tracing of B-spline surfaces. CAD **19** (1987) 122-130
91. J. Yen, S. Spach, M Smith and R. Pulleyblank: Parallel boxing in B-spline intersection. IEEE Comp. Graph. and Appl. **11** (1991) 72-79

Graph Voronoi Regions for Interfacing Planar Graphs

Thomas Kämpke and Matthias Strobel
Forschungsinstitut für anwendungsorientierte Wissensverarbeitung FAW
Helmholtzstr. 16, 89081 Ulm, Germany
{kaempke,mstrobel}@faw.uni-ulm.de

Abstract

Commanding motion is supported by a touch screen interface. Human input demonstrating trajectories by a sequence of points may be incomplete, distorted etc. Theses e ects are compensated by a transformation of vertex sequences of a regular grid into paths of a planar graph which codes feasible motions. The transformation is based on alteration operations including re-routings and on a so-called graph Voronoi regions which partition the plane according to proximity to vertices and edges.

Keywords: graph Voronoi region, grid graph, touch screen.

1 Introduction

Touch screen specifications of routes in a graph are investigated for the Euclidean space. A graph is therefore overlayed with a regular grid. The interplay between the graph and the grid gives rise to a variety of questions such as how to transform a sequence of grid points into a (meaningful) path in the graph. This task is similar to raster vector conversion with the difference that "vectors" cannot be chosen arbitrarily here but have to be taken from the graph. There is no true or ultimate transformation here since the intended path may adhere to ergonomical, aesthetic, or other criteria. Subsequent solutions should hence be considered as elements that may be combined in different manners.

The motivation for this problem stems from non-keyboard man-machine interfaces. Dynamic pointing operations typically serve for moving a scroll bar or an icon and for obtaining artistic effects from drawing with digital ink [2, p. 13]. Here, dynamic pointing relates to visible structures that restrict real motion in analogy to "streets". Applications of the approach include methods for input to spatial planning systems like navigation systems and techniques for commanding mobile systems by allowing explicit human guidance.

The difficulty of grid to graph transformations stems from the regular neighbourhoods of the grid and the irregular neighbourhoods of the graph being indepen-

V.N. Alexandrov et al. (Eds.): ICCS 2001, LNCS 2073, pp. 708–717, 2001.
©Springer-Verlag Berlin Heidelberg 2001

dent of each other. The transformations operate on two levels where the lower level utilizes geometric concepts while the upper level operates by production rules for regular expressions on mixed sequences of vertices and edges. So-called graph Voronoi regions will account for proximity towards vertices *and* edges.

2 Graphs, Grids, and Their Relation

Undirected graphs $G = (V, E)$ with vertex set V and edge set E are assumed to be simple meaning that there is at most one edge between any two vertices and no edge connects a vertex with itself (no loops). Each edge e is labeled by a non-negative length $c(e) = c_e$. A path is a vertex sequence with successive vertices being adjacent. As the graph is simple, successive vertices of a path are distinct but revisiting vertices is feasible within a path. The cost of a path $P(v, w) = (v = v_1, \ldots, v_s = w)$ from v to w with $\{v_1, v_2\}, \ldots, \{v_{s-1}, v_s\} \in E$ is $c(P(v, w)) := \sum_{i=1}^{s-1} c(v_i, v_{i+1})$. A shortest path from v to w is denoted $P_0(v, w)$. The degenerate case $P_0(v, v)$ is the single vertex v.

All graphs are planar, connected, and the length of a vertex sequence that need not be a path is considered later. The length of such a sequence is defined as sum of labels of successive vertices with cost assignment $d(v, u) := c(v, u)$ for $\{v, u\} \in E$ and $d(v, u) := c(P_0(v, u))$ for $\{v, u\} \not\in E$.

Actual point sets connecting vertices are denoted by $cur(v_1, v_2)$. Edges and curves are symmetric in the sense that $e = \{v_1, v_2\} = \{v_2, v_1\}$ and $cur(v_1, v_2) = cur(v_2, v_1)$. Curves may have several intersection points but no common sections.

Grids are unbounded and consist of equidistant horizontal and vertical lines. Each grid point has eight neighbours which are reachable by moving along lines and diagonals until the next point. The neighbourhood set of grid point p is $N(p)$ and the extended neighbourhood includes the grid point itself, i.e. $N'(p) = N(p) \cup \{p\}$.

A vertex with smallest Euclidean distance towards a grid point p is $v(p) = argmin_{v \in V} ||v - p||_2$ and a grid point with smallest Euclidean distance towards graph vertex v is $p(v) = argmin_{p \in P} ||p - v||_2$. The grid is assumed to be finer than the graph meaning that distinct graph vertices lie apart by at least the grid width. Thus, distinct graph vertices have distinct closest grid points. The Voronoi region of a vertex is the set of all points with smaller distance to that vertex than to any other vertex, $V(v) = \{x \,|\, ||x - v|| \leq ||x - w|| \; \forall\, w \in V - \{v\}\}$ with v being the *center* of the Voronoi region. When clear from the context, Voronoi regions will consist only of the grid points contained in the proper Voronoi regions.

3 Sequence Transformations

A grid point sequence $\bar{p} = (p^{(1)}, \ldots, p^{(N)})$ induces the sequence of closest graph vertices $\bar{v}(\bar{p}) = (v(p^{(1)}), \ldots, v(p^{(N)}))$. The grid point sequence is connected if each grid point is an extended neighbour of its predecessor. Even a connected

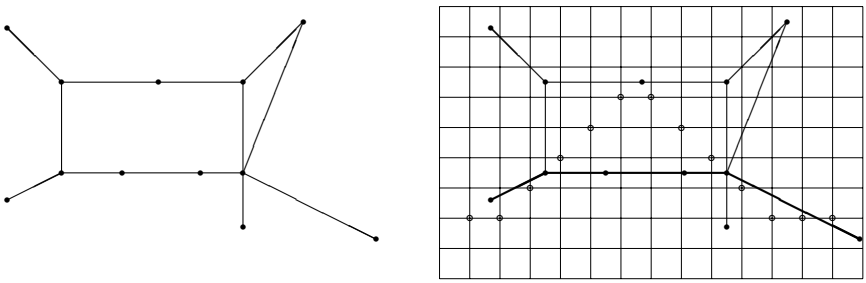


Figure 1: When specifying the grid point sequence (white dots) for a path (bold edges), the grid and the Voronoi regions are invisible (left).

grid point sequence need not induce a path. This property is addressed by forming traces which indicate the changes in vertex sequences. The trace of sequence $(v^{(1)}, \ldots, v^{(M)})$ with $v^{(j_1)} = v^{(1)} = v^{(2)} = \ldots = v^{(j_2-1)} \not= v^{(j_2)} \ldots \not= v^{(j_M)} = v^{(M)}$ is the subsequence $tr(v^{(1)}, \ldots, v^{(M)}) = (v^{(j_1)}, \ldots, v^{(j_M)})$. An example is $tr(v_4, v_4, v_3, v_4, v_5, v_5, v_4, v_7) = (v_4, v_3, v_4, v_5, v_4, v_7)$. Whenever the trace is a path, this path is taken as the transform of the grid point sequence. In other cases, vertex insertions and deletions are required.

3.1 Isolated Insertions and Deletions

A vertex sequence which is not a path can be transformed into a path by vertex insertions between successive vertices with $\{v^{(i)}, v^{(i+1)}\} \not\in E$. Vertex insertions will be obtained from shortest paths and they may adhere to additional constraints such as not using vertices from the present sequence or from other insertions. Decisions on allowing vertex repetitions ultimately appear to be possible only by convention rather than by purely geometrical criteria. Vertex deletion may serve as an alternative to insertions but this cannot be guaranteed to result in a path.

3.2 Joint Insertions and Deletions

Joint insertions and deletions also known as indels from string editing [3] adhere to connectivity. Therefore, a vertex of a sequence is understood to be isolated from the sequence, if the vertex is neither joined to its predecessor nor to its successor. A vertex that is not isolated from a sequence is connected to that sequence. A vertex $v^{(i)}$ is understood to be a single isolated vertex if $\{v^{(i-2)}, v^{(i-1)}\} \in E$, $\{v^{(i-1)}, v^{(i)}\} \not\in E$, $\{v^{(i)}, v^{(i+1)}\} \not\in E$, and $\{v^{(i+1)}, v^{(i+2)}\} \in E$.

A vertex sequence receives insertions so that a single isolated vertex $v^{(i)}$ becomes connected if it lies on at least one shortest path from $v^{(i-1)}$ to $v^{(i+1)}$. Otherwise the vertex is deleted and again a shortest path is inserted from $v^{(i-1)}$ to $v^{(i+1)}$, compare figures 2 and 3.

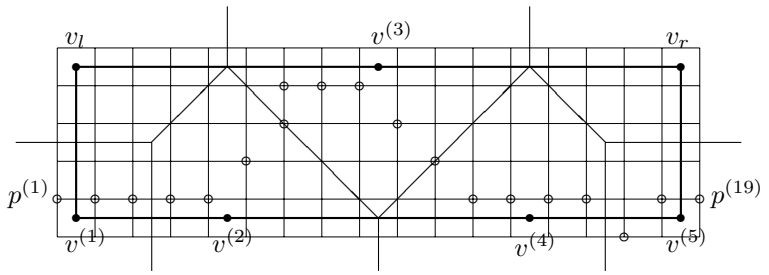


Figure 2: Graph with edges given by bold lines. Thin lines specify the boundaries of the Voronoi regions. Vertex $v^{(3)}$ is singly isolated in $(v^{(1)}, \ldots, v^{(5)})$ which is induced by the connected sequence $(p^{(1)}, \ldots, p^{(19)})$ indicated by white dots.

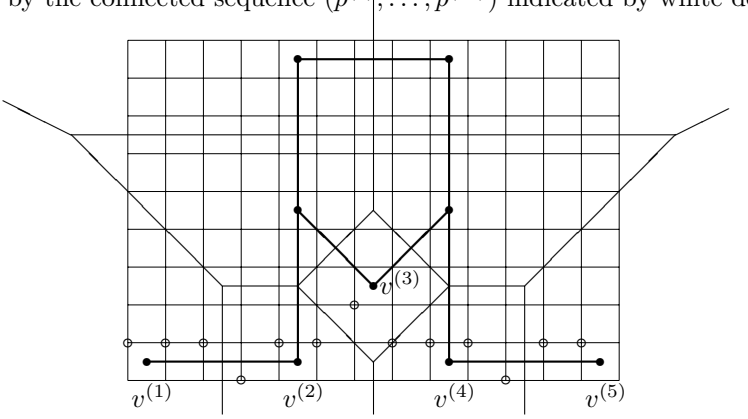


Figure 3: Vertex $v^{(3)}$ again is a singly isolated. As it is located on the unique shortest path from $v^{(2)}$ to $v^{(4)}$, it becomes connected to the vertex sequence.

3.3 Graph Voronoi Regions for Planar Graphs

Vertex proximity as expressed by Voronoi regions is not suitable for path specification. The reason is that closely following an edge which traverses a Voronoi region but is not incident with that region's center suggests an unintended vertex visit, compare figure 4. An appropriate partition of the plane is offered by forming certain Voronoi regions within Voronoi regions. These are based on the distance between a set A and a point x with respect to a Voronoi region:

$$dist_{V(v)}(x, A) := \begin{cases} \inf_{a \in A \cap V(v)} ||x - a||, & \text{for } A \cap V(v) \neq \emptyset \\ \infty, & \text{for } A \cap V(v) = \emptyset. \end{cases}$$

The **graph Voronoi regions** are established to express proximity to any graph element. Whenever a point from an ordinary Voronoi region is closest to that region's center or to an edge incident with the center, the point's assignment to the Voronoi region remains unchanged. Whenever a point from an ordinary

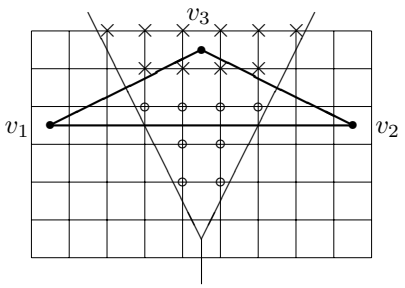


Figure 4: A connected grid point sequence following closely the edge $\{v_1, v_2\}$ leads to vertex v_3 being included in the induced vertex sequence. The mixed Voronoi region $V_{V(v_3)}(cur(v_1, v_2))$ contains all grid points marked by white dots. The remaining pure Voronoi region contains the grid points marked by crosses.

Voronoi region is closest to an edge that is not incident with the center of the Voronoi region, that edge receives a subset of the Voronoi region and the point under consideration is assigned to that subset. This results in the subsequent definitions of pure and mixed (graph) Voronoi regions.

$$
\begin{aligned}
V_{V(v)}(v) \;&:=\; \{x \in V(v) |\; \exists\, cur(v, v_i) \text{ such that } dist_{I\!R^2}(x, cur(v, v_i)) \leq \\
&\qquad dist_{I\!R^2}(x, cur(v_k, v_l)), \\
&\qquad \forall\, cur(v_k, v_l) \text{ with } v_k, v_l \in V - \{v\}\} \\
V_{V(v)}(cur(v_i, v_j)) \;&:=\; \{x \in V(v) |\; dist_{I\!R^2}(x, cur(v_i, v_j)) \leq dist_{I\!R^2}(x, cur(v_k, v_l)), \\
&\qquad \forall\, cur(v_k, v_l) \text{ with } \{v_k, v_l\} \neq \{v_i, v_j\}\} \text{ for} \\
&\qquad v_i, v_j \in V - \{v\}.
\end{aligned}
$$

A pure Voronoi region coincides with the ordinary Voronoi region if and only if all its mixed Voronoi regions are empty. For graph Voronoi regions a curve traversing an ordinary Voronoi region without being incident with the center affects grid points of this region in the same way as a curve that passes by.

A grid point from a pure Voronoi region $V(v)$ or $V_{V(v)}(v)$ will induce vertex v and a grid point from a mixed Voronoi region $V_{V(v)}(cur(v_i, v_j))$ will induce the edge $\{v_i, v_j\}$. Formally, for any $p \in P$

$$
ind(p) := \begin{cases} v, & p \in V(v) \text{ or } V_{V(v)}(v) \text{ for some } v \\ \{v_i, v_j\}, & p \in V_{V(v)}(cur(v_i, v_j)) \text{ for some } v \text{ and } v_i, v_j \in V - \{v\}. \end{cases}
$$

3.4 From Mixed Sequences to Paths

3.4.1 Operations on Sequences

A sequence of vertices and edges is called a mixed sequence. The mixed sequence induced by a grid point sequence $\bar{p} = (p^{(1)}, \ldots, p^{(N)})$ is denoted by $ind(\bar{p}) = (ind(p^{(1)}), \ldots, ind(p^{(N)}))$. The trace of a mixed sequence is understood in analogy

to the trace of a vertex sequence. Mixed sequences are transformed to vertex sequences according to a set $\mathcal{O}$ of operations. Their specification is based on strings such as $A[x], A[x,y]$ etc. which denote possibly empty strings like $A[x] = x, x$ and $A[x,y] = y, y, x, y$. $A_1[x], A_1[x,y]$ etc. denote strings that consist of at least one of the bracketed terms. The vertex sequence resulting from no further operation of $\mathcal{O}$ being applicable is denoted by $\bar{v}(\cdot)$.

$O1$. $(X, v^{(i)}, A[v^{(i)}, \{v^{(i)}, v^{(i+1)}\}], v^{(i+1)}, Y) \rightarrow (X, v^{(i)}, v^{(i+1)}, Y)$
for $last(X) \not= v^{(i)}$ and $first(Y) \not= v^{(i+1)}$.

$O2$. $(X, v^{(i)}, A[v^{(i)}, \{v^{(i)}, v^{(i+1)}\}], A[\{v^{(i+1)}, v^{(i+2)}\}], v^{(i+2)}, Y)$
$\rightarrow (X, v^{(i)}, v^{(i+1)}, v^{(i+2)}, Y)$
for $last(X) \not= v^{(i)}$ and $first(Y) \not= v^{(i+2)}$.

$O3$. $(X, A_1[v^{(i)}, \{v^{(i)}, v^{(i+1)}\}], \ldots, A_1[v^{(i+k)}, \{v^{(i+k)}, v^{(i+k+1)}\}], v^{(i+k+1)}, Y)$
$\rightarrow (X, v^{(i)}, \ldots, v^{(i+k+1)}, Y)$
for $last(X) \not= v^{(i)}, \{v^{(i)}, v^{(i+1)}\}$
and $first(Y) \not= v^{(i+k+1)}, \{v^{(i+k)}, v^{(i+k+1)}\}$; $k \geq 0$.

$O4$. $(X, A_1[v^{(i)}, \{v^{(i)}, v^{(i+1)}\}], \ldots, A_1[v^{(i+k)}, \{v^{(i+k)}, v^{(i+k+1)}\}],$
$A_1[v^{(i+k+2)}, \{v^{(i+k+1)}, v^{(i+k+2)}\}], Y) \rightarrow (X, v^{(i)}, \ldots, v^{(i+k+2)}, Y)$
for $last(X) \not= v^{(i)}, \{v^{(i)}, v^{(i+1)}\}$ and $first(Y) \not= v^{(i+k+2)}$; $k \geq 0$.

$O5$. $tr(ind(\bar{p})) = (X, S_1, \ldots, S_k, Y)$
$\rightarrow (X, in(S_1), out(S_1), \ldots, in(S_k), out(S_k), Y)$
for select components $S_1, \ldots, S_k$, $k \geq 1$
v being the last vertex to which X is transformed or $X = \epsilon$
$w = first(Y)$ or $Y = \epsilon$
and no other operation applicable; see text.

The prefix X and the suffix Y may be the empty string ϵ. In case several successive edges neither share a vertex with their predecessor nor with their successor, operations $O1$ - $O4$ may not be applicable or may result in multiple sequence ambiguities. Such cases are resolved by select components. A select component of a mixed sequence is defined to be a $\subseteq$-maximal subsequence of successive vertices and edges such that it is either a single vertex, or a single edge, or applications of $O1$ - $O4$ lead to a unique vertex sequence.

Each select component has an entry vertex and an exit vertex which is unique in case the select component is a single vertex or leads to a unique vertex sequence. Otherwise, these vertices admit a twofold ambiguity. In the unique case, the entry vertex and the exit vertex may be identical as for a single vertex or a complete cycle. Unique entry and exit vertices are denoted by $v(in, S_i)$ and $v(out, S_i)$, the others are denoted by $v(in, S_i, 1)$, $v(in, S_i, 2)$, $v(out, S_i, 1)$, and $v(out, S_i, 2)$, where $v(in, S_i, 1) = v(out, S_i, 2)$ and $v(in, S_i, 2) = v(out, S_i, 1)$. Ambiguities are resolved by forming shortest paths as in figure 5. All edges receive $d(\cdot, \cdot)$ labels with $d(v, \cdot)$ and $d(\cdot, w)$ becoming zero in case $X = \epsilon$ and $Y = \epsilon$ respectively.

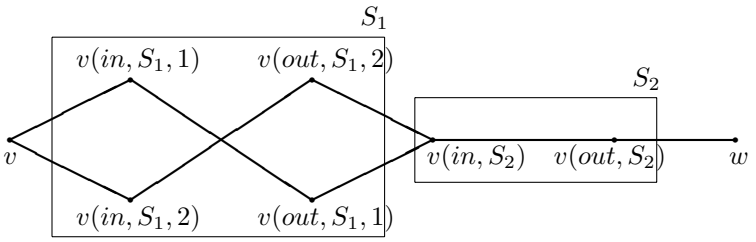


Figure 5: Substitution graph for shortest paths through S_1 and S_2 where S_1 has non-unique entry and exit vertices while S_2 has unique entry and exit vertices.

3.4.2 Transformations

Whenever a vertex sequence results from the operations of $\mathcal{O}$ applied to a mixed sequence, the vertex sequence has no immediate repetition. The transformation based on the extended definition of $\bar{v}$ formally is again given by

$$Tr(\bar{p}) := \begin{cases} \bar{v}(tr(ind(\bar{p}))), & \text{if } \bar{v}(tr(ind(\bar{p}))) \text{ is a path in } G \\ \text{void}, & \text{else.} \end{cases}$$

Whenever the trace of an induced sequence alternates between two vertices such as (v_4, v_2, v_4, v_2), no operation of $\mathcal{O}$ applies and thus the sequence is left unchanged by $\bar{v}(\cdot)$. It is thus possible to state deliberate vertex repetitions in paths by suitable grid point sequences.

3.4.3 Complete Transformations

In case $Tr(\bar{p})$ is void, the vertex sequence $\bar{v}(tr(ind(\bar{p})))$ can be extended to a path by inserting shortest paths between any successive vertices that are not adjacent in G. A reasonable decision on inserting and deleting vertices can be based on the connectivity of the grid point sequence. If the grid point sequence is disconnected, deletions are forbidden. The reason is that a disconnection of the grid point sequence may result from deliberate jumps to sections of the graph that must be visited by the path. If the grid point sequence is connected, single isolated vertices will be deleted if they do not lie on a shortest connecting path in G, otherwise they will be connected. The complete procedure is as follows.

A1

1. Input $\bar{p}$ with $Tr(\bar{p}) = \bar{v}(tr(ind(\bar{p}))) = (v^{(1)}, \ldots, v^{(M)}) = \bar{v}$.
2. **If** $Tr(\bar{p})$ is a path, no operations are performed;
   else **if** $\bar{p}$ is disconnected then any $v^{(i)}, v^{(i+1)}$ with $\{v^{(i)}, v^{(i+1)}\} \not\in E$ are connected by $P_0(v^{(i)}, v^{(i+1)})$ giving new path $\bar{v}$;
   else insertions $P_0(v^{(i)}, v^{(i+1)})$, $P_0(v^{(i+1)}, v^{(i+2)})$ are replaced by $P_0(v^{(i)}, v^{(i+2)})$ if $v^{(i+1)}$ is single isolated vertex in original $\bar{v}$ giving new $\bar{v}$.
3. Output path $\bar{v}$.

Whenever vertex repetitions in the final path are unintended they can be suppressed by best shortenings [5]. If connectivity of the grid point sequence is no criterion of the path construction, potential deletion of a single isolated vertex can still be considered meaningful giving the next algorithm.

A2

1. Input $\bar{p}$ with $Tr(\bar{p}) = \bar{v}(tr(ind(\bar{p}))) = (v^{(1)}, \ldots, v^{(M)}) = \bar{v}$.
2. **If** $Tr(\bar{p})$ is a path, no operations are performed;
   else any $v^{(i)}, v^{(i+1)}$ with $\{v^{(i)}, v^{(i+1)}\} \not\in E$ are connected by $P_0(v^{(i)}, v^{(i+1)})$ giving new path $\bar{v}$, insertions $P_0(v^{(i)}, v^{(i+1)})$ and $P_0(v^{(i+1)}, v^{(i+2)})$ are replaced by $P_0(v^{(i)}, v^{(i+2)})$ if $v^{(i+1)}$ is single isolated vertex in original $\bar{v}$.
3. Output path $\bar{v}$.

4 Computational Issues

The computation of graph Voronoi regions can be reduced to computing Voronoi regions of vertices and of a finite collection of line segments and then taking certain intersections of these regions. Both individual computations can be perforemd in $O(n \log n)$, see [9] for the latter, but these computations are concepturally complicated. They have even lead to approximations of Voronoi regions of a finite collection of line segments by angular bisector regions [1].

A simple approximation of graph Voronoi regions relies on the nearest vertex and the nearest edge for each grid point being computable in $O(n)$; planarity of the graph ensures that it has at most $3n - 6$ edges. Whenever the nearest line is incident with the nearest vertex, the grid point lies in the pure Voronoi region of that vertex. Otherwise, the grid point lies in the mixed Voronoi region of that line with respect to the nearest vertex. Computing the mixed induced sequence of a grid point sequence of legth N then requires time $O(Nn)$.

The previous approximations of graph Voronoi regions and transformations by algorithm **A2** were implemented with input obtained from an elo 151R IntelliTouch 15-inch touch screen. Geometric data handling was organized within the LEDA system, and the algorithms were written in C++.

Figures 6 through 9 show a planar graph and a sequence of about 200 grid points being transformed to a path. Edges are labeled by their Euclidean lengths. Figure 8 depicts approximations of the graph Voronoi regions. The "overshoot" (U-shaped section) of the grid point sequence in figure 7 in the center region of the graph is so large that vertex insertions occur.

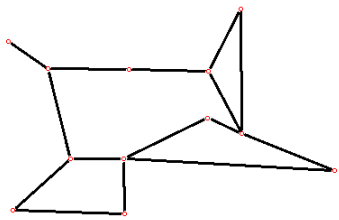


Figure 6: Planar graph with 12 vertices.

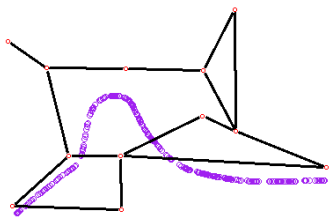


Figure 7: Graph with grid point sequence, grid omitted.

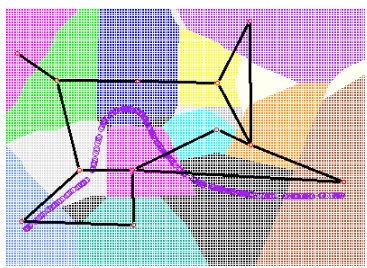


Figure 8: Graph, grid point sequence, and approximated graph Voronoi regions.

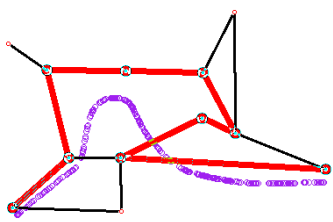


Figure 9: Graph with grid point sequence transformed to a path (bold edges).

References

[1] Cloppet, F., Olivia, J.-M., Stamon, G., "Angular bisector network, a simplified generalized Voronoi diagram: applications to processing complex intersections in biomedical images", IEEE Transactions on Pattern Analysis and Machine Intelligence 22, 2000, p. 120-128.

[2] Cohen, P. et al., "Multimodal interaction for 2D and 3D environments", Computer Graphics and Applications, July/August 1999, p. 10-13.

[3] Gusfield, D., "Algorithms on strings, trees, and sequences: computer science and computational biology", Cambridge University Press, Cambridge, 1997.

[4] Hopcroft, J.E., Ullman, J., "Introduction to automata theory, languages and computation", Addison-Wesley, New York, 1979.

[5] Kämpke, T., "Interfacing graphs", Journal of Machine Graphics and Vision 9, 2000, p. 797-824.

[6] Leeuwen, J.v. (ed.), "Handbook of theoretical computer science: algorithms and complexity", vol. A, Elsevier, Amsterdam, 1990.

[7] O'Rourke, J., "Computational Gemetry in C", 2nd ed., Cambridge University Press, Cambridge, 1998.

[8] Sugihara, K., "Approximations of generalized Voronoi diagrams by ordinary Voronoi diagrams", Computer Vision and Graphic Image Processing 55, 1993, p. 522-531.

[9] Yap, C.K., "An $O(n * \log n)$ algorithm for Voronoi diagrams of a set of simple curve segments", Discrete and Computational Geometry 2, 1987, p. 365-393.

Robust and Fast Algorithm for a Circle Set Voronoi Diagram in a Plane

Deok-Soo Kim[1], Donguk Kim[1], Kokichi Sugihara[2], and Joonghyun Ryu[1]

[1] Department of Industrial Engineering, Hanyang University
17 Haengdang-Dong, Sungdong-Ku, Seoul, 133-791, Korea
dskim@email.hanyang.ac.kr
{donguk, jhryu}@cadcam.hanyang.ac.kr

[2] Department of Mathematical Engineering and Information Physics, Graduate School of Engineering, University of Tokyo, 7-3-1, Hongo, Bunkyo-ku, Tokyo, 113, Japan
sugihara@simplex.t.u-tokyo.ac.jp

Abstract. Robust and fast computation of the exact Voronoi diagram of circle set is difficult. Presented in this paper is an edge-flipping algorithm that computes a circle set Voronoi diagram using a point set Voronoi diagram, where the points are the centers of circles. Hence, the algorithm is as robust as its counterpart of point set. Even though the theoretical worst-case time complexity is quadratic, the actual performance shows a strong linear time behavior for various test cases. Furthermore, the computation time is comparable to the algorithm of point set Voronoi diagram itself.

1 Introduction

Let $\mathbf{P} = \{\mathbf{p}_i \mid i = 1, 2, \ldots, n\}$ be the set of the centers $\mathbf{p}_i$ of circles $\mathbf{c}_i$ in a plane, and $\mathbf{C} = \{\mathbf{c}_i \mid i = 1,2,...,n\}$ be the set of circles $\mathbf{c}_i = (\mathbf{p}_i, r_i)$ where r_i is the radius of $\mathbf{c}_i$. VD($\mathbf{P}$) and VD($\mathbf{C}$) are the Voronoi diagrams for $\mathbf{P}$ and $\mathbf{C}$, respectively. Suppose that we want to compute the exact VD($\mathbf{C}$) where the radii of possibly intersecting circles are not necessarily equal. Several researches exist on this or related problems. Lee and Drysdale first considered Voronoi diagram for a set of non-intersecting circles [13], and suggested an O($n\log^2 n$) algorithm. They also reported another algorithm of $O(nc^{\sqrt{\log n}})$ [1,2]. Sharir reported an algorithm computing VD($\mathbf{C}$) in O($n\log^2 n$), where the circles may intersect [18]. Yap reported an O($n\log n$) time algorithm for line segments and circles [23]. While all of the above algorithms are based on the divide-and-conquer scheme, Fortune devised an O($n\log n$) time algorithm based on line sweeping [4]. Recently, Gavrilova and Rokne reported an algorithm to maintain the correct topology data structure of VD($\mathbf{C}$) when circles are dynamically moving [5]. Sugihara reported an approximation algorithm for VD($\mathbf{C}$) by sampling several points on the circles and computing the Voronoi diagram of these points [19].

In this paper, we present an algorithm that computes the Voronoi diagram of circle set correctly, robustly and efficiently. The robustness issue is the most important concern in this paper. The principle idea is as the following. Given a correct point set

V.N. Alexandrov et al. (Eds.): ICCS 2001, LNCS 2073, pp. 718-727, 2001.
© Springer-Verlag Berlin Heidelberg 2001

Voronoi diagram of the centers of circles, we compute the correct topology of VD(**C**) by flipping the edges of VD(**P**) of the centers. Then, we compute the equations of Voronoi edges.

It turns out that this approach works quite well. Since our approach is computing the correct topology of VD(**C**) by changing the topology of VD(**P**), our algorithm is as robust as a point set Voronoi diagram algorithm, provided that the decision can be made correctly. Note that the theory on the robustness issue for the point set Voronoi diagram has been well-established. Even though the theoretical worst-case time complexity is quadratic, the actual performance shows a strong linear time behavior for various test cases. In addition, the algorithm is quite simple to implement. The *edge* and *vertex* will be used in this paper to mean a Voronoi edge and a Voronoi vertex.

In this paper, we assume that the degrees of vertices of VD(**P**) as well as VD(**C**) are three, and VD(**P**) is represented in an efficient data structure such as a winged-edge data structure [14,17] and available a priori by a robust code such as [20,21,22] which is based on the exact computation strategy [6,20]. We also assume that the algorithm to compute the circumcircle(s) of three given circles, which is discussed in another paper [11], is available.

2 Edge Flipping

When an edge **e** in Fig. 1(a) is changed to to **e** in Fig. 1(b), it is called that **e** is *flipped* to **e** . Hence a flipping operation changes the pointers among the vertices, edges and generators appropriately.

As shown in Fig. 2, there are three possible configurations of an edge of VD(**P**) for the flipping test. An edge of VD(**P**) may have either two circumcircles, only one circumcircle or no circumcircle at the vertices of the edge. Fig. 2 shows the cases. When a circumcircle does not exist at a vertex, an inscribing circle actually exists at the given configuration.

2.1 Case I : Two Circumcircles

In Fig. 3, there are two vertices $\mathbf{v}_1$ and $\mathbf{v}_2$ on an edge $\mathbf{e}_1$. Let $\mathbf{CC}_i$ be a circumcircle about three generators corresponding to a vertex $\mathbf{v}_i$. When $\mathbf{e}_1$ is considered, the

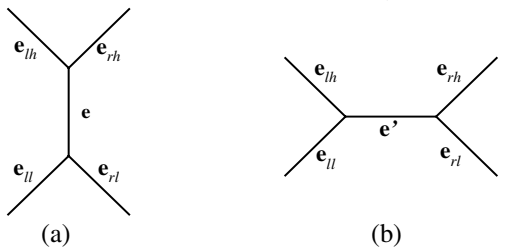


Fig. 1. Topology configuration after an edge flipping operation

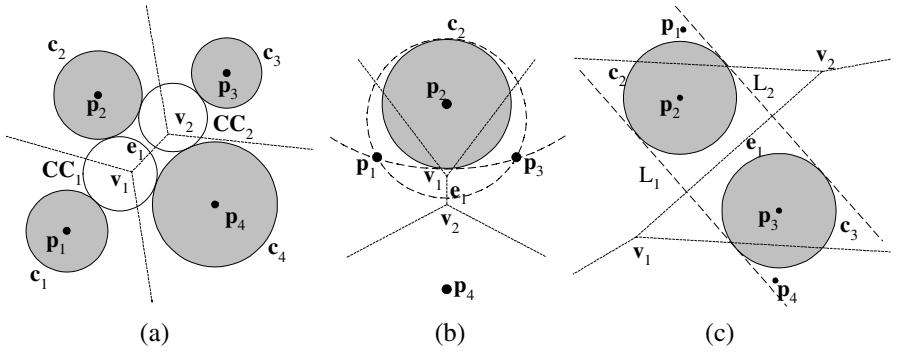


Fig. 2. Edge configurations

generator $\mathbf{c}_3$ is called a mating generator of $\mathbf{CC}_1$ and denoted as $\mathbf{M}_1$. When there exist circumcircles at the both ends of an edge, the circumcircles may or may not intersect with their mates.

Lemma 1. If both circumcircles do not intersect with their mates, the edge should not flip.

Proof. (Fig. 3(a)) The edge e_1 of VD(**P**) shown with dotted lines has two vertices $\mathbf{v}_1$ and $\mathbf{v}_2$. The vertex $\mathbf{v}_1$ has three associated generators $\mathbf{p}_1$, $\mathbf{p}_2$ and $\mathbf{p}_4$, and the vertex $\mathbf{v}_2$ has three associated generators $\mathbf{p}_3$, $\mathbf{p}_4$ and $\mathbf{p}_2$. Let $\mathbf{CC}_1$ be a circumcircle to three circles $\mathbf{c}_1$, $\mathbf{c}_2$ and $\mathbf{c}_4$. From the definition of vertex $\mathbf{v}_1$ of VD(**P**), it can be determined that $\mathbf{CC}_1$ should be computed from $\mathbf{c}_1$, $\mathbf{c}_2$ and $\mathbf{c}_4$. Similarly, $\mathbf{CC}_2$ is a circumcircle to $\mathbf{c}_3$, $\mathbf{c}_4$ and $\mathbf{c}_2$. Note that we call $\mathbf{c}_3$ a mating generator of $\mathbf{CC}_1$. Since $\mathbf{CC}_1 \quad \mathbf{c}_3 =$ in the figure, any point inside or on $\mathbf{CC}_1$ is closer to $\mathbf{c}_1$, $\mathbf{c}_2$ and $\mathbf{c}_4$ than any point on $\mathbf{c}_3$. Similarly, $\mathbf{CC}_2 \quad \mathbf{c}_1 =$, and any point on $\mathbf{CC}_2$ is closer to $\mathbf{c}_2$, $\mathbf{c}_3$ and $\mathbf{c}_4$ than $\mathbf{c}_1$. Since same property holds for the centers of circles, the topology of VD(**P**) should be identical to the topology of VD(**C**). Therefore, the topology of VD(**P**) can be correctly used for the topology of VD(**C**) without any modification.

Lemma 2. If both circumcircles intersect with their mates, the edge should flip.

Proof. (Fig. 3(b)) The point set is identical to Fig. 3(a) and the radii of the circles are different. Note that both $\mathbf{CC}_1$ and $\mathbf{CC}_2$ intersect with mates $\mathbf{c}_3$ and $\mathbf{c}_1$, respectively. The fact that $\mathbf{CC}_1$ intersects with the mate $\mathbf{c}_3$ means that $\mathbf{c}_3$ has a point on the circle which is closer to the vertex $\mathbf{v}_1$ than any point on the three associated circles $\mathbf{c}_1$, $\mathbf{c}_2$ and $\mathbf{c}_4$. This suggests that the topology of vertex $\mathbf{v}_1$, as was given in VD(**P**), cannot exist as a member of vertex set in VD(**C**). Similarly, the vertex $\mathbf{v}_2$ cannot be a member of vertex set of VD(**C**), since $\mathbf{CC}_2$ also intersects with $\mathbf{c}_1$. Therefore, the edge $\mathbf{e}_1$ cannot exist in VD(**C**) as the topological structure given in VD(**P**) because both end vertices of the edge should disappear simultaneously. On the other hand, $\mathbf{c}_1$, $\mathbf{c}_2$, and $\mathbf{c}_3$ define a valid new vertex $\mathbf{v}_1$ and $\mathbf{c}_1$, $\mathbf{c}_4$, and $\mathbf{c}_3$ define another valid vertex $\mathbf{v}_2$. Topologically connecting $\mathbf{v}_1$ and $\mathbf{v}_2$ with an edge creates a new Voronoi edge $\mathbf{e}_1$. Therefore, a new edge $\mathbf{e}_1$ should be born while the old edge $\mathbf{e}_1$ disappears, and this results in an edge flipping.

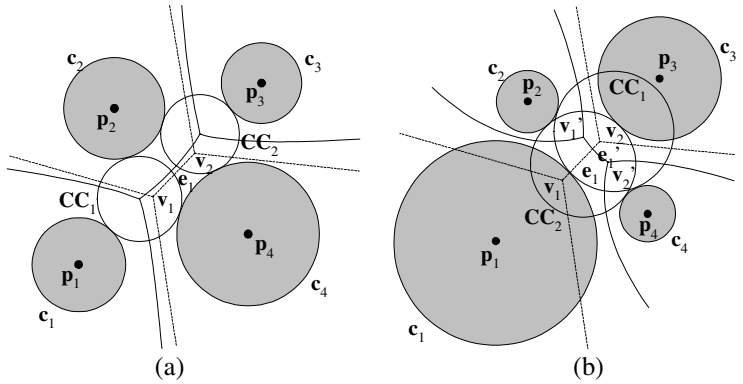


Fig. 3. Point sets in both figures (a) and (b) are identical, and therefore the point set Voronoi diagrams (shown with dotted lines) are identical as well. However, the corresponding circle set Voronoi diagrams (shown with solid curves) differ.

Between two circumcircles, it is possible that only one circumcircle intersects with its mating generator. Suppose that the circumcircles are $\mathbf{CC}_1$ and $\mathbf{CC}_2$ corresponding to $\mathbf{v}_1$ and $\mathbf{v}_2$, respectively. Let $\mathbf{CC}_1$ $\mathbf{M}_1$ and $\mathbf{CC}_2$ $\mathbf{M}_2 =$. Since $\mathbf{CC}_1$ $\mathbf{M}_1$, the topology of vertex $\mathbf{v}_1$ should be changed in the topology update process, while the topology of $\mathbf{v}_2$ should remain as it was given since $\mathbf{CC}_2$ $\mathbf{M}_2 =$. Because of this small conflict, the current edge cannot be directly flipped. However, this conflict can be resolved by flipping another edge incident to the vertex $\mathbf{v}_1$ in a later step so that the topological structure of $\mathbf{v}_1$ is valid, while the topology of $\mathbf{v}_2$ remains at this moment. This observation provides the following lemma.

Lemma 3. If one circumcircles intersects with its mates, the edge should not flip.

2.2 Case II : One Circumcircle

Lemma 4. If one circumcircle exists, and the circumcircle intersects with its mate, the edge should flip.

Proof. (Fig. 4) As shown in Fig. 4(b) there is a case that no circumcircle, corresponding to vertex $\mathbf{v}_1$, exists to three generators $\mathbf{p}_1$, $\mathbf{c}_2$, and $\mathbf{p}_3$. Note that both dotted circles, associated to vertex $\mathbf{v}_1$, in the figure are not valid circumcircles, but circles inscribing $\mathbf{c}_2$. The fact that there is no circumcircle to three generator circles means the Voronoi vertex of three generator circles should disappear. In the given case, on the other hand, a circumcircle corresponding to vertex $\mathbf{v}_2$ exists and the circumcircle intersects with the mating generator $\mathbf{c}_2$. When this phenomenon happens an edge $\mathbf{e}_1$ should flip to $\mathbf{e}_1$.

Even though a circumcircle exists, it is possible that the existing circumcircle does not intersect with the mating generator circle. Obviously, the edge should not flip in this case and therefore the following lemma results.

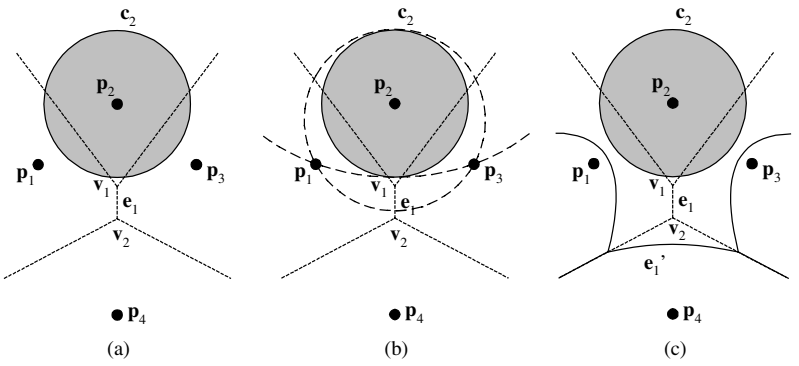


Fig. 4. A case that only one circumcircle exists and the existing circumcircle intersects with the mating generator.

Lemma 5. Only one circumcircle exists, and the circumcircle does not intersect with its mate, the edge should not flip.

2.3 Case III : No Circumcircle

It is even possible that an edge does not yield any valid circumcircles. Instead, only inscribing circles are defined by the circle generators. In this case, the edge does not flip as stated by the following lemma.

Lemma 6. When no circumcircle exists, the edge should not flip.

3 Special Cases Due to Convex Hull

While the above six lemmas guarantee the robust and correct transformation from the topology of VD(**P**) to that of VD(**C**), there could be a few special cases that need careful treatment. Let CH(**A**) be the convex hull of set **A**.

A flipping operation does not change the cardinality of topology while the generator stays inside of CH(**P**). Since it is viewed as a continual process, there could be an incident that four generators are circumcircular and one edge disappears. However, it is assumed that this case, which can be handled by extending the proposed algorithm, does not exist in this paper.

As the radius of a generator increases, a number of interesting and tricky problems may occur. The edges and vertices of VD(**P**) may sometimes disappear, and new edges and vertices, which were not in VD(**P**), are created when certain conditions are satisfied. Both cases, which have a direct relationship with the convex hulls of both generator sets, are elaborated in this section.

Similarly to a Voronoi diagram of a point set, a Voronoi region of $\mathbf{c}_i$ of VD(**C**) is infinite if and only if $\mathbf{c}_i$ CH(**C**) . Due to this observation, a Voronoi region defined by generators interior to CH(**C**) always defines a bounded region. Since

CH(**P**) and CH(**C**) may have different generators in their boundaries, there may be changes of bounded and unbounded regions in both Voronoi diagrams. This process involves the changes of the cardinality as well as the structure of the topology of Voronoi diagrams.

Suppose that a point **p** was a vertex of CH(**P**), and located interior to CH(**C**). Then, as will be discussed soon, one unbounded region of VD(**P**) becomes a bounded one in VD(**C**). This causes changes in the number of vertices and edges, too. The number of edges is increased by one, and so is the number of vertices. Similar phenomenon exists in the opposite case. In other words, when a point **p** was interior to CH(**P**) and the circle **c**, which corresponds to **p**, intersects with the boundary of CH(**C**), a bounded region now becomes an unbounded infinite region and creates one new vertex as well as one new edge. If there is no change between the generator sets that lie on the boundaries of CH(**P**) and CH(**C**), the number of edges, and therefore vertices, of VD(**C**) is identical to that of VD(**P**). The details of these cases related to the convex hulls are not discussed here.

4 Edge Geometry

Once the topology of VD(**C**) is fixed, it is necessary to compute the edge equations of VD(**C**) to complete the construction. The equation of a Voronoi edge of Voronoi diagram of circles is either a part of line or hyperbola [2,10]. The cases of parabolic and elliptic arcs do not occur in our problem. Persson and Held represented the edge equations using a parametric curve obtained by solving the intersection equations of the offset elements of generators [8,9,16]. In their representation, both line and hyperbola are represented in different forms. On the other hand, Kim used a rational quadratic Bézier curve to represent the edges. In this representation, any type of bisectors, for example, line, parabola, hyperbola, or ellipse, can be represented in a unified form, and hence used in this paper, too.

It is known that a conic arc can be converted into a rational quadratic Bézier curve form which is defined as

$$(t) = \frac{w_0(1 \quad t)^2\mathbf{b}_0 + 2w_1t(1 \quad t)\mathbf{b}_1 + w_2t^2\mathbf{b}_2}{w_0(1 \quad t)^2 + 2w_1t(1 \quad t) + w_2t^2} \qquad t \quad [0,1] \tag{1}$$

where $\mathbf{b}_0$, $\mathbf{b}_1$ and $\mathbf{b}_2$ are the control points, and w_0, w_1 and w_2 are the corresponding weights. It is known that a rational quadratic Bézier curve (t) representation of a conic curve can be computed if two points $\mathbf{b}_0$ and $\mathbf{b}_2$ on the curve, two tangents of the curve at $\mathbf{b}_0$ and $\mathbf{b}_2$, and another passing point **p** on the curve are known [3]. Among these five conditions, two points $\mathbf{b}_0$ and $\mathbf{b}_2$ are already known since the bisector should pass through two vertices of a Voronoi edge. Another passing point on the bisector can be obtained trivially as a point on the line segment defined by the centers of two generator circles and equidistant from two circles. Two last conditions of tangent vectors can be obtained by the following lemma which can be proved without much difficulty.

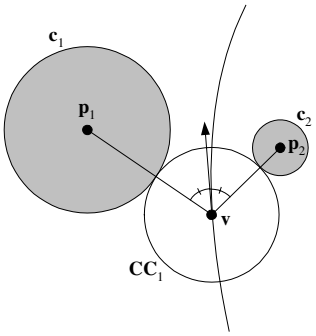
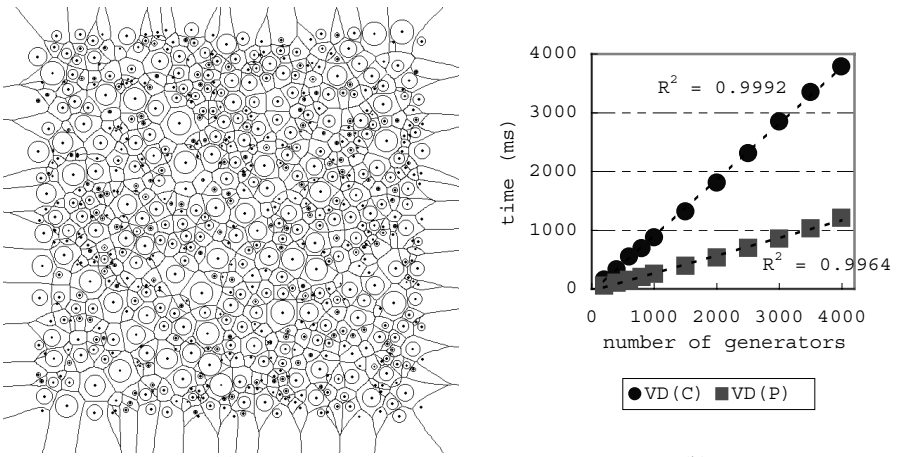


Fig. 5. A tangent vector on a bisector.

Lemma 7. Let a bisector (t) be defined between two circles $\mathbf{c}_1$ and $\mathbf{c}_2$. Then, the tangent line of (t) at a point $\mathbf{v}$ is given by an angle bisector of $\mathbf{p}_1\mathbf{v}\mathbf{p}_2$, where $\mathbf{p}_1$ and $\mathbf{p}_2$ are the centers of $\mathbf{c}_1$ and $\mathbf{c}_2$, respectively.

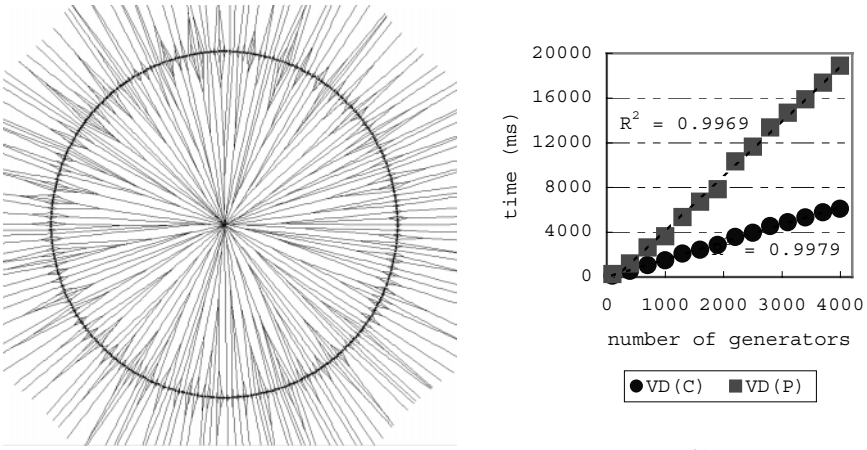
5 Implementation and Experiments

The proposed algorithm has been implemented and tested on MSVC++ on Intel Celeron 300MHz processor. Fig. 6 and Fig. 7 show two examples. In Fig. 6, 800 random circles are generated and do not intersect each other and have different radii. And In Fig. 7, the 400 non-intersecting circles with different radii generated on a large circle. Fig. 6(a) and Fig. 7(a) show results, and Fig. 6(b) and Fig. 7(b) show the computation time taken by a number of generator sets with varying cardinality. In the figure, the computation time taken by a code to compute the Voronoi diagram of point sets is denoted by VD(**P**), and the time taken by our code to compute the Voronoi



(a) (b)

Fig. 6. (a) Voronoi diagram of 800 random circles. (b) The computation time taken to compute the Voronoi diagram of point sets, VD(**P**), and our code to compute the Voronoi diagram of circle sets, VD(**C**).



(a) (b)

Fig. 7. (a) Voronoi diagram of 400 random circles on a large circle. (b) The computation time taken by a code to compute the Voronoi diagram of point sets, VD(**P**), and our code to compute the Voronoi diagram of circle sets, VD(**C**).

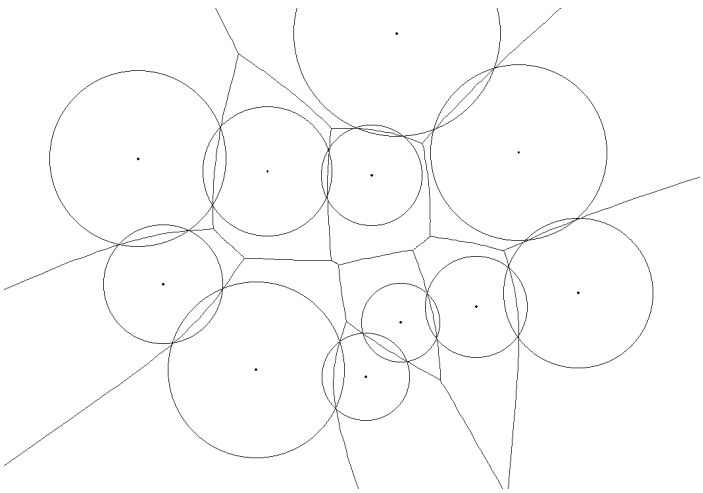


Fig. 8. An example when the generators intersect each other.

diagram of circle sets is denoted by VD(**C**). The point sets are the centers of circles generated at random, in this example. Note that the time denoted by VD(**C**) does not include the time taken by a preprocessing, which is actually the time denoted by VD(**P**). Therefore, the actual computation time to compute VD(**C**) from a given circle set is the accumulation of both computation times.

Comparing VD(**C**) with VD(**P**), it can be deduced that VD(**C**) is not as big as it might have been expected. Through experiences, there are cases that VD(**C**) is even much smaller than VD(**P**). Also, note that the correlation coefficient shown in the figure suggests that the average running behavior is a strong linear one. We have experimented with many other cases, and all the cases shows similar linear pattern.

Based on these experiments we claim that the proposed algorithm is very efficient and robust. Even though the worst-case scenario, which will given $O(n^2)$ time performance, is theoretically possible, it is difficult to expect to face such a case in reality. Fig. 9 shows that our algorithm works for the cases that the circles intersect each other. Fig. 9a shows the result of preprocessing which is the Voronoi diagram of point set, and 9b shows the Voronoi diagram of circle set.

6 Conclusions

Presented in this paper is an algorithm to compute the exact Voronoi diagram of circle set from the Voronoi diagram of point set. Even though the time complexity of the proposed algorithm is $O(n^2)$, the algorithm is quite fast, produces exact result, and robust.

The algorithm uses the point set Voronoi diagram of the centers of circles as an initial solution, and finds the correct topology of the Voronoi diagram of circle set by flipping the appropriate edges of the point set Voronoi diagram. Then, the edge equations are computed. Because our algorithm uses a point set Voronoi diagram, which has been studied extensively in its robustness as well as performance, the proposed algorithm is as robust as a point set Voronoi diagram.

Acknowledgements

The first author was supported by Korea Science and Engineering Foundation (KOSEF) through the Ceramic Processing Research Center (CPRC) at Hanyang University, and the third author was supported by Torey Science Foundation, Japan.

References

1. Drysdale, R.L.III, Generalized Voronoi diagrams and geometric searching, Ph.D. Thesis, Department of Computer Science, Tech. Rep. STAN-CS-79-705, Stanford University, Stanford CA (1979).
2. Drysdale, R.L.III, and Lee, D.T, Generalized Voronoi diagram in the plane, Proceedings of the 16th Annual Allerton Conference on Communications, Control and Computing, Oct. (1978) 833-842.
3. Farin, G., Curves and Surfaces for Computer-Aided Geometric Design: A Practical Guide, 4th edition, Academic Press, San Diego (1996).
4. Fortune, S., A sweepline algorithm for Voronoi diagrams, Algorithmica, Vol. 2 (1987) 153-174.
5. Gavrilova, M. and Rokne, J., Swap conditions for dynamic Voronoi diagram for circles and line segments, Computer Aided Geometric Design, Vol. 16 (1999) 89-106.
6. Gavrilova, M., Ratschek, H. and Rokne, J., Exact computation of Delaunay and power triangulations, Reliable Computing, Vol. 6 (2000) 39-60.

7. Hamann, B. and Tsai, P.-Y., A tessellation algorithm for the representation of trimmed NURBS surfaces with arbitrary trimming curves, Computer-Aided Design, Vol. 28, No. 6/7 (1996) 461-472.
8. Held, M., On the Computational Geometry of Pocket Machining, LNCS, Springer-Verlag (1991).
9. Held, M., Lukács, G. and Andor, L., Pocket Machining Based on Contour-Parallel Tool Paths Generated by Means of Proximity Maps, Computer-Aided Design, Vol.26, No. 3 (1994) 189-203.
10. Kim, D.-S., Hwang, I.-K. and Park, B.-J., Representing the Voronoi diagram of a simple polygon using rational quadratic Bézier curves, Computer-Aided Design, Vol. 27, No. 8 (1995) 605-614.
11. Kim, D.-S., Kim, D., Sugihara, K., Ryu, J., Apollonius tenth problem as a Point Location Problem, (Submitted to ICCS 2001).
12. Kim, D.-S., Kim, D., and Sugihara, K., Voronoi diagram of a circle set from Voronoi diagram of a point set: II. Geometry, (Submitted to Computer Aided Geometric Design).
13. Lee, D.T. and Drysdale, R.L.III, Generalization of Voronoi diagrams in the plane, SIAM J. COMPUT., Vol. 10, No. 1, February (1981) 73-87.
14. Mäntylä, M., An introduction to solid modeling, Computer Science Press (1988).
15. Okabe, A., Boots, B. and Sugihara, K., Spatial Tessellations Concepts and Applications of Voronoi Diagram, John Wiley & Sons (1992).
16. Persson, H., NC machining of arbitrarily shaped pockets, Computer-Aided Design, Vol. 10, No. 3 (1978) 169-174.
17. Preparata, F.P. and Shamos, M.I. Computational Geometry An Introduction Springer-Verlag (1985).
18. Sharir, M., Intersction and closest-pair problems for a set of planar discs, SIAM J. COMPUT., Vol. 14, No. 2 (1985) 448-468.
19. Sugihara, K., Approximation of generalized Voronoi diagrams by ordinary Voronoi diagrams, Graphical Models and Image Processing, Vol. 55, No. 6 (1993) 522-531.
20. Sugihara, K., Experimental study on acceleration of an exact-arithmetic geometric algorithm, Proceedings of the IEEE International Conference on Shape Modeling and Applications (1997) 160-168.
21. Sugihara, K. and Iri, M., Construction of the Voronoi diagram for one million generators in single-precision arithmetic, Proc. IEEE 80 (1992) 1471-1484.
22. Sugihara, K., `http://www.simplex.t.u-tokyo.ac.jp/~sugihara/`, (2000).
23. Yap, C.K., An O(*n*log*n*) algorithm for the Voronoi diagram of a set of simple curve segments, Discrete Comput. Geom., Vol. 2 (1987) 365-393.

Apollonius Tenth Problem as a Point Location Problem

Deok-Soo Kim[1], Donguk Kim[1], Kokichi Sugihara[2], and Joonghyun Ryu[1]

[1] Department of Industrial Engineering, Hanyang University
17 Haengdang-Dong, Sungdong-Ku, Seoul, 133-791 Korea
dskim@email.hanyang.ac.kr
{donguk, jhryu}@cadcam.hanyang.ac.kr
[2] Department of Mathematical Engineering and Information Physics, Graduate School of Engineering, University of Tokyo, 7-3-1, Hongo, Bunkyo-ku, Tokyo, 133 Japan
sugihara@simplex.t.u-tokyo.ac.jp

Abstract. Given a set of three circles in a plane, we want to find a circumcircle to these given circles called generators. This problem is well known as Apollonius Tenth Problem and is often encountered in geometric computations for CAD systems. This problem is also a core part of an algorithm to compute the Voronoi diagram of circles. We show that the problem can be reduced to a simple point-location problem among the regions bounded by two lines and two transformed circles. The transformed circles are produced from the generators via linear fractional transformations in a complex space. Then, some of the lines tangent to these transformed circles corresponds to the desired circumcircle to the generators. The presented algorithm is very simple yet fast. In addition, several degenerate cases are all incorporated into one single general framework.

1 Introduction

Suppose that we want to compute circumcircles of a set of three circles in a plane. The radii of the circles are assumed to be not necessarily equal and where the circles are possibly intersecting one another. This problem is frequently encountered in various geometric computations in CAD systems and the computation of the Voronoi diagram of circles [3,8,10,11,13,15,18].

The problem can be solved in various ways. One approach could be computing the center of circumcircle as an intersection between two bisectors defined by two circles. It turns out that this process involves the solution process of a quartic equation that can be solved by either the Ferrari formula or a numerical process [9]. Note that this approach can be applied only after the number of circumcircles to the generators is determined. On the other hand, the solution may be symbolically generated via tools like Mathematica. However, the cost of such symbolic generation can be also quite high.

It is known that there are at most eight circles simultaneously tangent to three circle generators as shown in Fig. 1. In this and the following figures, the black circles are given generator circles while the white ones are tangent circles.

Among the tangent circles, we want to find the circumcircles for three generator circles. Depending on the configuration of three generators, however, there may be

V.N. Alexandrov et al. (Eds.): ICCS 2001, LNCS 2073, pp. 728-737, 2001.
© Springer-Verlag Berlin Heidelberg 2001

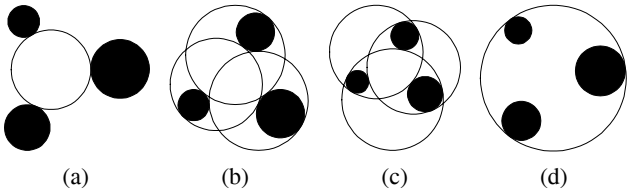


Fig. 1. Circles tangent to three generator circles

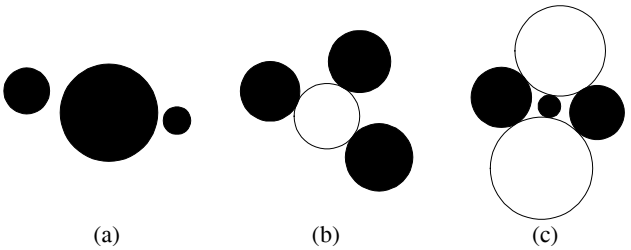


Fig. 2. Circumcircles. (a) no circumcircle exists, (b) one circumcircle exists, and (c) two circumcircles exist.

either no, one, or two circumcircles, as shown in Fig. 2. We want to determine which case a given generator set is and find such circumcircles with less computation as possible if they exist.

In Section 2, we provide the previous researches related to the problem. In Section 3, the properties of the linear fractional transformation in a complex plane are provided so that the problem can be transformed to easier one. The discussions in this section is a slight variation of the noble approach initially presented by Rokne[16]. Based on the transformation, we present the point location problem formulation of the problem in Section 4.

2 Related Works

In his book *On Contacts*, Apollonius of Perga(262-190 B.C.), known as *The Great Geometer*, left the famous Apollonius problems : Given any three points, lines, or circles, or any combination of three of these, to construct a circle passing through the points and tangent to the given lines and circles. Among ten possible combinations of the geometric entities involved, The Apollonius' Tenth Problem is the most general problem to construct the circles simultaneously tangent to three circles. [2,4,5].

There have been several efforts to solve the problem in various ways [1,3,14,17]. Recently, Rokne reported an approach based on the linear fractional transformation (also known as Möbius transformation) in the complex plane [16]. Using the fact that a linear fractional transformation in a complex plane maps circles to lines and vice versa, he suggested to compute a tangent line of two circles in a mapped space to back-transform into a circumcircle. Most recently, Gavrilova reported an analytic solution which involves trigonometric functions [7].

Even though the problem is quite complicated in Euclidean space, it turns out that it can be rather easily solved by employing a complex system. Following Rokne's suggestion, we have adopted the linear fractional transformation to transform the given problem into the problem of finding tangent lines of two circles in a mapped space. Then, we formulate a point location problem so that all of the degenerate configurations of generators can be handled in a unified way. It turns out that our approach incorporate all variations of degeneracies in a single framework, is easy to program, numerically robust, and computationally very efficient. Hence the proposed algorithm is preferable for the implementation for geometric computations.

3 Linear Fractional Transformations

Let the plane, where the circles are given, be complex. Then, a point (x, y) in the Euclidean plane can be treated as a complex number $z = x + iy$. Also, let $\mathbf{c}_i = (z_i, r_i)$, i = 1, 2, and 3, be the generator circles with a center (x_i, y_i) and a radius $r_1 \geq r_2 \geq r_3 \geq 0$ as shown in Fig. 3. Then, $\tilde{\mathbf{c}}_i = (z_i, r_i - r_3)$ transforms generator circles $\mathbf{c}_1$, $\mathbf{c}_2$ and $\mathbf{c}_3$ to shrunk circles $\tilde{\mathbf{c}}_1$, $\tilde{\mathbf{c}}_2$ and $\tilde{\mathbf{c}}_3$ respectively. Note that $\tilde{\mathbf{c}}_3$ degenerates to a point z_3. Then, if we can find a circle $\tilde{\mathbf{c}}$ passing through $z_3 \equiv \tilde{\mathbf{c}}_3$ and tangent to both $\tilde{\mathbf{c}}_1$ and $\tilde{\mathbf{c}}_2$, we can easily find a circle $\mathbf{c}$ which is simultaneously tangent to $\mathbf{c}_1$, $\mathbf{c}_2$ and $\mathbf{c}_3$ by simply subtracting r_3 from the radius of $\tilde{\mathbf{c}}$.

Consider a linear fractional transformation defined as

$$W(z) = \frac{az + b}{cz + d} \tag{1}$$

where $ad - bc \neq 0$, and a, b, c and d are either complex or real numbers. Note that $W(z)$ is analytic so that the mapping $W(z)$ is everywhere conformal and maps circles and straight lines in the Z-plane onto circles and straight lines in the W-plane. Among others, we note a particular linear mapping

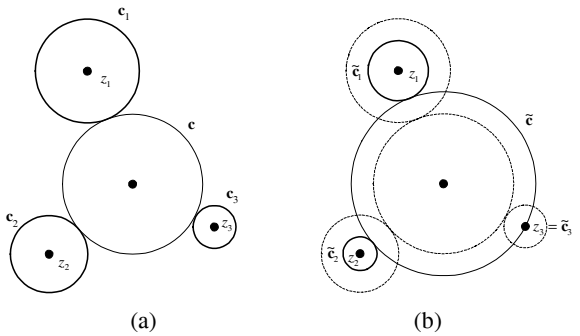


Fig. 3. Circumcircle and the inflated circumcircle. (a) generators and the desired circumcircle, (b) shrunk generators and a circumcircle passing through z_3.

$$W(z) = \frac{1}{z - z_0} \tag{2}$$

as was suggested by [6,16]. The mapping defined in Equation (2) is known to possess the following properties.

- It transforms lines and circles passing through z_0 in the *Z*-plane to straight lines in the *W*-plane.
- It transforms lines and circles not passing through z_0 in the *Z*-plane to circles in the *W*-plane.
- It transforms a point at infinity in the *Z*-plane to the origin of the *W*-plane.

The details can be found in a material on the subject such as [12]. Therefore, a mapping $W(z) = 1 / (z - z_3)$ transforms $\tilde{\mathbf{c}}_1$ and $\tilde{\mathbf{c}}_2$ in the *Z*-plane to circles W_1 and W_2 in the *W*-plane, if z_3 is not on $\tilde{\mathbf{c}}_1$ and $\tilde{\mathbf{c}}_2$. Then, the desired circle $\tilde{\mathbf{c}}$ tangent to circles $\tilde{\mathbf{c}}_1$ and $\tilde{\mathbf{c}}_2$ in the *Z*-plane will be mapped to a line *L* tangent to W_1 and W_2 in the *W*-plane by $W(z)$. It can be shown that $W(z)$ maps circles $\tilde{\mathbf{c}}_i = (z_i, r_i - r_3)$ into circles $W_i = (\omega_i, R_i)$ defined as

$$\omega_i = \left(x_i - x_3/D_i, -\left(y_i - y_3\right)/D_i\right) \tag{3}$$

$$R_i = \left(r_i - r_3\right)/D_i$$

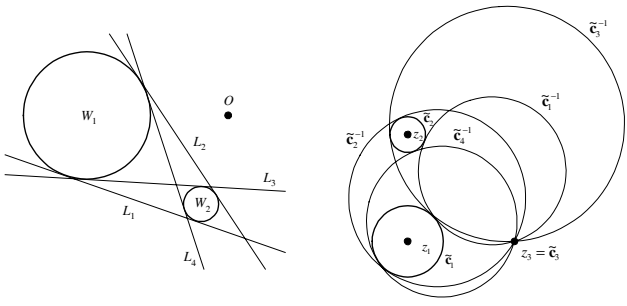
where $D_i = (x_i - x_3)^2 + (y_i - y_3)^2 - (r_i - r_3)^2$, i = 1 and 2. Similarly, it can be also shown that the inverse transformation

$$W^{-1}(z) = Z(w) = 1/w + z_3 \tag{4}$$

is also another conformal mapping, and hence, maps lines not passing through the origin of the *W*-plane to circles in the *Z*-plane. Suppose that a line is given as $au + bv + 1 = 0$ in the *W*-plane. Then, its inverse in the *Z*-plane is a circle $\tilde{\mathbf{c}} = (z_0, r_0)$, where $z_0 = \left(-a/2 + x_3, b/2 + y_3\right)$ and $r_0 = \sqrt{a^2 + b^2}/2$. We recommend [16] for the details of the computation using this mapping.

4 Point Location Problem

Based on Rokne's approach to transform Z-plane to *W*-plane, we formulate the problem as a point location problem. Let W_1 and W_2 be two circles with radii R_1 and R_2 in the *W*-plane, respectively. Suppose that $R_1 > R_2 > 0$, as shown in Fig. 4(a). Then, there could be at most four distinct lines simultaneously tangent to both W_1 and W_2. Suppose that the black dot in Fig. 4(a) is the origin *O* of the coordinate system in the *W*-plane. Then, the line L_1 maps to the circumcircle $\tilde{\mathbf{c}}_1^{-1}$ in the *Z*-plane, as shown in Fig. 4(b), by the inverse mapping $Z(w)$ because the circles W_1 and W_2 as well as the origin *O* are located in the same side with respect to L_1. Since the origin *O* of the *W*-plane corresponds to infinity in the *Z*-plane and $Z(w)$ is conformal, $\tilde{\mathbf{c}}_1$ and $\tilde{\mathbf{c}}_2$ in the *Z*-plane are located to the infinity from the inverse mapped circle $\tilde{\mathbf{c}}_1^{-1}$ and therefore $\tilde{\mathbf{c}}_1^{-1}$



(a) (b)

Fig. 4. $W^{-1}(z) = Z(w) = 1/w + z_3$ maps from the *W*-plane to the *Z*-plane. (a) the *W*-plane, (b) the *Z*-plane.

should be the desired circumcirlce. Therefore, we can also derive an observation of $O \notin (W_1 \cup W_2)$, which means that the origin *O* of the *W*-plane cannot lie on or interior to the circles W_1 and W_2.

Similarly, L_2 maps to the inscribing circle $\tilde{\mathbf{c}}_2^{-1}$ since the circles W_1 and W_2 are in the opposite side of *O* which corresponds to the infinity in the *Z*-plane. Cases of L_3 and L_4 correspond to $\tilde{\mathbf{c}}_3^{-1}$ and $\tilde{\mathbf{c}}_4^{-1}$, respectively. Therefore, the line *L* which corresponds to a circumcircle in the Z-plane is either one or both of the *exterior tangent lines*, L_1 and/or L_2. Between L_1 and L_2, the one containing W_1, W_2 and the origin *O* in the same side of the line will map to the desired circumcircle(s). Remember that zero, one or both exterior tangent lines may be the correct result depending on the configuration of the initially given generator circles. From now on, we will drop the word *exterior* from the term for the convenience of presentation, unless otherwise needed.

4.1 Decomposition of the *W*-Plane

Suppose W_1 and W_2, $R_1 > R_2 \neq 0$, are given as shown in Fig. 5(a). Let L_1 and L_2 be the tangent lines to both circles. Let L_i^+ be the half-space, defined by L_i, containing W_1 as well as W_2. Similarly, L_i^- means the opposite side of L_i^+. Then, *W*-plane consists of six mutually exclusive regions as follows:

$$\alpha = \left(L_1^+ \cap L_2^-\right) \cup \left(L_1^- \cap L_2^+\right) \qquad \beta = L_1^- \cap L_2^- \qquad \gamma = L_1^+ \cap L_2^+$$
$$\delta = L_1 \cap L_2 \qquad \varepsilon = \left(L_1 \cap L_2^-\right) \cup \left(L_1^- \cap L_2\right) \qquad \zeta = \left(L_1 \cap L_2^+\right) \cup \left(L_1^+ \cap L_2\right)$$

As shown in the figure, the region α consists of two subregions and the region γ consists of three (or four, if W_1 and W_2 intersect each other) subregions.

4.2 Location of the Origin of the *W*-Plane

Once the *W*-plane is decomposed into a set of such regions, the problem of computing a circumcircle(s) now further reduces to a point location problem among the regions.

Note that, in Fig.5, the shaded circles are shrunk circles, and black dots are the shrunk circles with zero radii and thus degenerate to a point in the Z-plane. In addition, a circumcircle is shown in a solid curve while an inscribing circle is shown in a broken curve.

Theorem 1. If $R_1 > R_2 \neq 0$, there are six cases as follows.

- Case α: If $O \in \alpha$, one tangent line maps to a circumcircle while the other tangent line maps to an inscribing circle. (Fig. 5(b)-α)
- Case β: If $O \in \beta$, both tangent lines map to inscribing circles. (Fig. 5(b)-β)
- Case γ: If $O \in \gamma$, both tangent lines map to circumcircles. (Fig. 5(b)-γ)
- Case δ: If $O \equiv \delta$, both tangent lines map to lines intersecting at a point. (Fig. 5(b)-δ)
- Case ε: If $O \in \varepsilon$, a tangent line on which O lies maps to a line, while the other tangent line maps to an inscribing circle. (Fig. 5(b)-ε)
- Case ζ: If $O \in \zeta$, the tangent line on which O lies maps to a line, while the other tangent line maps to a circumcircle. (Fig. 5(b)-ζ)

Proof.

- Case α: Suppose that $\alpha_1 = \left(L_1^- \cap L_2^+\right)$ and $\alpha_2 = \left(L_1^+ \cap L_2^-\right)$. Without loss of generality we can assume that $O \in \alpha_1$. Then, L_1 in the W-plane is inverse-mapped to a circle $\tilde{\mathbf{c}}_1^{-1}$ inscribing $\tilde{\mathbf{c}}_1$ and $\tilde{\mathbf{c}}_2$ in the Z-plane, as illustrated by a dotted curve in Fig.5(b)-α. This is because L_1 places O on the opposite side of W_1 and W_2. Note that $\tilde{\mathbf{c}}_1$ and $\tilde{\mathbf{c}}_2$ are the inverse maps of W_1 and W_2. On the other hand, L_2 is inverse-mapped to a circumcircle $\tilde{\mathbf{c}}_2^{-1}$ tangent to $\tilde{\mathbf{c}}_1$ and $\tilde{\mathbf{c}}_2$ in the Z-plane, and is illustrated as a solid curve. This is because L_2 places W_1, W_2 and O on the same side. Since two tangent lines in the W-plane intersect each other at δ, the inverse mapped circles, regardless they are circumcircles or inscribing circles, always intersect each other at $W^{-1}(\delta)$ computed by Eq.(4) shown as a black rectangle in the Z-plane.
- Case β: When $O \in \beta$, both W_1 and W_2 are on the opposite side of O with respect to both tangent lines L_1 and L_2. Therefore, both L_1 and L_2 should be mapped to inscribing circles, and hence, no circumcircle will result as shown in Fig. 5(b)-β.
- Case γ: When $O \in \gamma$, both W_1 and W_2 are on the same side of O with respect to both tangent lines L_1 and L_2. Hence, both L_1 and L_2 should be mapped to circumcircles only. In this case, two different situations may occur. Note that the region γ consists of three subreigons. The case in Fig. 5(b)-γ_1 occurs when O lies in-between two circles W_1 and W_2, and the case γ_2 in Fig. 5(b)-γ_2 occurs when O lies in the other subregions of γ.
- Case δ: When $O \equiv \delta$, the inverse mapping to the Z-plane yields results similar to what is shown in the W-plane. Since the tangent lines in W-plane pass through the origin O, the inverse-mapped (supposedly) circles should pass through the infinity. This means that the radii of the inverse-mapped circles are infinite. Therefore, the mapping results in lines in Z-plane as shown in Fig. 5(b)-δ. Note that they only intersect at $\tilde{\mathbf{c}}_3$.

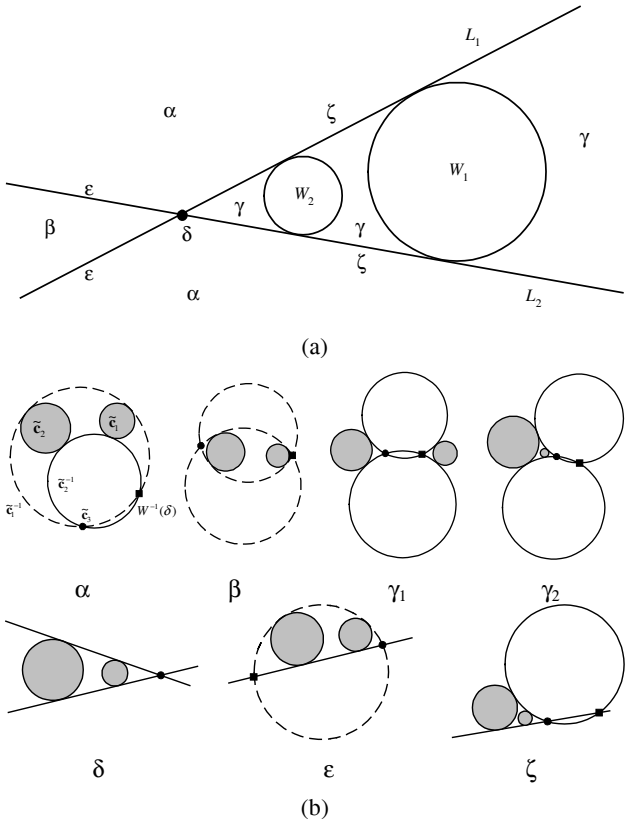


Fig. 5. $R_1 = R_2 > 0$. (a) the *W*-plane, (b) the *Z*-plane

- Case ε: When $O \in \varepsilon$, *O* lies precisely on a ray ε starting from δ. In this case, the corresponding tangent line on which *O* lies is inverse mapped to a line in the *Z*-plane, as was explained in the above. Then, *O* should be located on the opposite side of the other tangent line with respect to W_1 and W_2, meaning that there is an inscribing circle as shown in Fig. 5(b)-ε.
- Case ζ: When $O \in \zeta$, *O* lies precisely on a ray ζ, which is also a ray starting from δ. In this case, the corresponding tangent line inverse maps to another line in the *Z*-plane similarly to the above cases. In this case, however, *O* as well as W_1 and W_2 should be located on the same side of the other tangent line. It means that the tangent line inverse maps to a circumcircle in the *Z*-plane as shown in Fig. 5(b)-ζ.

Note that some tangent circles to shrunk circles degenerate to lines in Cases δ, ε and ζ. In this case, the desired tangent circles to the generators can be obtained by translating the degenerate lines to the opposite direction of the shrunk circles.

Slightly changing the configuration of generator circles, various degeneracies may occur. It turns out that the degeneracies are mainly due to the radii of W_1 and W_2.

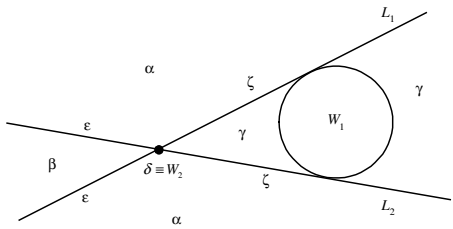


Fig. 6. $R_1 > R_2 = 0$.

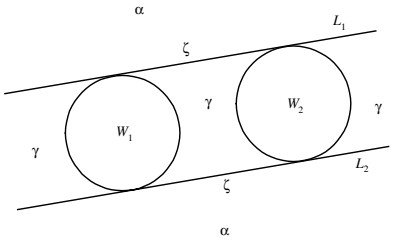


Fig. 7. $R_1 = R_2 > 0$

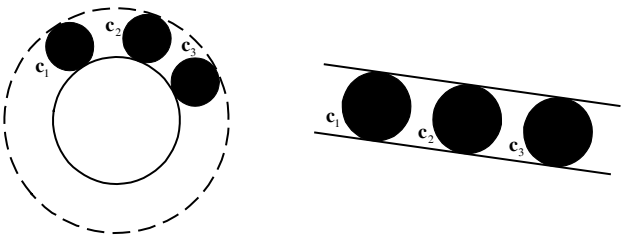


Fig. 8. $R_1 = R_2 = 0$: generator circles in the *Z*-plane

4.3 Degenerate Cases

Even though the problem has been discussed for a general case, there could be several degeneracies which may make the problem more difficult. The degeneracies are mainly due to the radii of W_1 and W_2. It turns out that, however, the theory previously discussed can be used for such degeneracies without much modifications.

One degenerate case is $R_1 > R_2 = 0$, which means that W_2 degenerates to a point as shown in Fig. 6. This case occurs when two smaller generator circles $\mathbf{c}_2$ and $\mathbf{c}_3$ in the *Z*-plane have identical radii. The differences of this case from the general case are the followings: i) The region γ consists of two subregions, and ii) Case δ does not occur. Otherwise, everything is same as before.

A second degenerate case occurs when $R_1 = R_2 > 0$, as shown in Fig. 7, which means that W_1 and W_2 have identical non-zero radii. Note that $R_1 = R_2$ in general does not guarantee $r_1 = r_2$, which are the radii of generator circles. In other words, even though two generator circles in the *Z*-plane have identical radii, the radii of mapped circles in the *W*-plane are not necessarily identical, and vice versa. Note that two exterior tangent lines in the *W*-plane are parallel in this case. In this case, therefore, the regions β, δ, and ε disappear. Therefore, the cases left are Cases α, γ, and ζ, and Theorem 1 still holds except the missing cases.

A third, and the last, degenerate case is $R_1 = R_2 = 0$, and is illustrated in Fig. 8. This case occurs when both W_1 and W_2 have zero radii, and therefore $L_1 \equiv L_2$. This case is possible only when all generator circles in the Z-plane have identical radii. In this case, only the regions α and ζ only remain. The interpretations of the remaining

regions stay the same as before. Note that Fig. 6 and Fig. 7 illustrate the *W*-plane while Fig. 8 shows the *Z*-plane.

Therefore, these degenerate cases can be all treated in a unified algorithm without any modification, except the minor treatment of parsing the regions. One possible special treatment would be the very last case where the centers of three circles with identical radii are collinear. In this case, there are no circumcircle but two tangent lines, as shown in Fig. 8.(b), and they can be only computed by translation of the computed line.

5 Conclusions

Presented in this paper is an algorithm to compute the circumcircles of a set of three generator circles in a plane. This problem is a part of the well-known Apollonius' Tenth Problem and is frequently encountered in various geometric computations for CAD systems as well as for the computation of the Voronoi diagram of circles. It turns out that this seemingly trivial problem is not an easy problem at all to solve in a general setting. In addition, there can be several degenerate configurations of the generators.

Even though the problem is quite complicated in Euclidean space, it turns out that it can be rather easily solved by employing a complex system. Following Rokne's approach, we have adopted the linear fractional transformation to transform the given problem into the problem of finding tangent lines of two circles in a mapped space. Then, we formulate a point location problem so that all of the degenerate configurations of generators can be handled in a unified way.

It turns out that the proposed approach incorporates all variations of degeneracies in a single framework, is easy to program, numerically robust, and computationally very efficient. We have also demonstrated the validity and efficiency of the algorithm by applying the theory to the computation of Voronoi diagram of circles.

We expect that the idea presented in this paper can extend to all Apollonius Problems to solve them in a single general framework, as far as the circumcircle is concerned.

Acknowledgements

The first author was supported by Korea Science and Engineering Foundation (KOSEF) through the Ceramic Processing Research Center (CPRC) at Hanyang University, and the third author was supported by Torey Science Foundation, Japan.

References

1. Altshiller-Court, N., The problem of Apollonius. College Geometry, 2nd Ed., Barnes and Noble, New York, (1952) 173-181.

2. Boyer, C. B., A History of Mathematics, Wiley, New York (1968).
3. Capelli, R. Circle tangential to 3 cicles or lines. Posting No. 35067, Usenet newsgroup comp.graphics.algorithms, 2 pages, (1996).
4. Courant, R. and Robbins, H., What is Mathematics?: An Elementary Approach to Ideas and Methods, 2nd edition, Oxford University Press, Oxford (1996).
5. Dörrie, H., 100 Great Problems of Elementary Mathematics: Their History and Solutions, Dover, New York (1965).
6. Gavrilova, M. and Rokne, J., Swap conditions for dynamic Voronoi diagram for circles and line segments, Computer Aided Geometric Design, Vol. 16 (1999) 89-106.
7. Gavrilova, M. and Rokne, J., Apollonius' Tenth Problem Revisited, Special Session on Recent Progress in Elementary Geometry, 941st American Mathematical Society Conference (1999) 64.
8. Kim, D.-S., Hwang, I.-K. and Park, B.-J., Representing the Voronoi diagram of a simple polygon using rational quadratic Bézier curves, Computer-Aided Design, Vol. 27, No. 8 (1995) 605-614.
9. Kim, D.-S., Lee, S.-W. and Shin, H., A cocktail algorithm for planar Bézier curve intersections, Computer-Aided Design, Vol. 30, No. 13 (1998) 1047-1051.
10. Kim, D.-S., Kim, D. and Sugihara, K., Voronoi diagram of a circle set from Voronoi diagram of a point set: I. Topology, (Submitted to Computer Aided Geometric Design 2001).
11. Kim, D.-S., Kim, D. and Sugihara, K., Voronoi diagram of a circle set from Voronoi diagram of a point set: II. Geometry, (Submitted to Computer Aided Geometric Design 2001).
12. Kreyszig, E., Advanced Engineering Mathematics, 7th Edition, John Wiley & Sons (1993).
13. Lee, D.T. and Drysdale, R.L., III, Generalization of Voronoi diagrams in the plane, SIAM J. COMPUT., Vol. 10, No. 1 (1981) 73-87.
14. Moise, E.E., Elementary Geometry from an Advanced Standpoint, 3rd. ed., Addison-Wesley Publ. Co., Reading (1990).
15. Okabe, A., Boots, B. and Sugihara, K., Spatial Tessellations Concepts and Applications of Voronoi Diagram, John Wiley & Sons (1992).
16. Rokne, J., Appolonius's 10th problem, Graphics Gems II, ed. James Arvo, Academic Press, (1991) 19-24.
17. Sevici, C.A., Solving the problem of Apollonius and other related problems, Graphics Gems III, ed. David Kirk, Academic Press, San Diego (1992) 203-209.
18. Sharir, M., Intersection and closest-pair problems for a set of planar discs, SIAM J. COMPUT., Vol. 14, No. 2 (1985) 448-468.

Crystal Voronoi Diagram and Its Applications to Collision-Free Paths

Kei Kobayashi[1] and Kokichi Sugihara[2]

[1] University of Tokyo, Hongo, Bunkyo-ku, Tokyo 113-8656, Japan,
kkoba@stat.t.u-tokyo.ac.jp
[2] University of Tokyo, Hongo, Bunkyo-ku, Tokyo 113-8656, Japan,
sugihara@simplex.t.u-tokyo.ac.jp

Abstract. This paper studies the multiplicatively weighted crystal-growth Voronoi diagram, which describes the partition of the plane into crystals with different growth speeds. This type of the Voronoi diagram is defined, and its basic properties are investigated. An approximation algorithm is proposed. This algorithm is based on a finite difference method, called a fast marching method, for solving a special type of a partial differential equation. The proposed algorithm is applied to the planning of a collision-free path for a robot avoiding enemy attacks.

1 Introduction

Suppose that various types of crystals grow from different start points in the plane with different speeds. When two crystal regions meet, they stop growing in that direction. Then, the plane is partitioned into individual crystal regions; this partition is called the *multiplicatively weighted crystal-growth Voronoi diagram*, which is the topic of this paper.

A number of types of generalized Voronoi diagrams have been proposed on the basis of different types of weighted distances, including the additively weighted Voronoi diagrams, the multiplicatively weighted Voronoi diagrams, and the compoundly weighted Voronoi diagrams [2,3]. However, the multiplicatively weighted crystal-growth Voronoi diagram is quite different from the others, because a crystal cannot enter into the area which is already occupied by another crystal. A crystal with a high speed should grow around avoiding slowly growing crystals. Hence, the "distance" between two points at a given time should be measured by the length of the shortest path that avoids crystal regions generated by that time. In this sense, the computation of this Voronoi diagram is very hard.

The concept of the multiplicatively weighted crystal-growth Voronoi diagram was first proposed by Schaudt and Drysdale [1]. They presented an $O(n^3)$ approximation algorithm for n crystals.

This paper studies this Voronoi diagram from various points of view. First, we present a new approximation algorithm for constructing this Voronoi diagram. Secondly, we apply this Voronoi diagram to the search of the shortest path for a robot that moves among enemy robots.

V.N. Alexandrov et al. (Eds.): ICCS 2001, LNCS 2073, pp. 738–747, 2001.
© Springer-Verlag Berlin Heidelberg 2001

The structure of the paper is the following. In Section 2, we review definitions and fundamental properties of Voronoi diagrams. In Section 3, we construct a new algorithm for approximately computing the multiplicatively weighted crystal-growth Voronoi diagram, and in Section 4, it is applied to the collision-free path planning for robots. In Section 5, we give the conclusion.

2 Multiplicatively Weighted Crystal-Growth Voronoi Diagram

2.1 Ordinary Voronoi Diagram

Let $S = \{\mathrm{P}_1, \mathrm{P}_2, \cdots, \mathrm{P}_n\}$ be a set of n points in the plane. For each P_i, let $R(S; \mathrm{P}_i)$ be the set of points that are nearer to P_i than to other P_j's ($j \neq i$), that is,

$$R(S; \mathrm{P}_i) = \{\mathrm{P} \mid \|\mathrm{P} - \mathrm{P}_i\| < \|\mathrm{P} - \mathrm{P}_j\|,\ j \neq i\}, \tag{1}$$

where $\|\mathrm{P} - \mathrm{Q}\|$ denotes the Euclidean distance between the two points P and Q. The plane is partitioned into $R(S; \mathrm{P}_1), R(S; \mathrm{P}_2), \cdots, R(S; \mathrm{P}_n)$ and their boundaries. This partition is called the *Voronoi diagram* for S, and the elements of S are called the *generators* of the Voronoi diagram. The region $R(S; \mathrm{P}_i)$ is called the *Voronoi region* of P_i, and the boundary lines of the Voronoi diagram are called *Voronoi edges*.

In the following subsections we generalize the concept of the Voronoi diagram. In order to avoid confusion, the above-defined Voronoi diagram is sometimes called the *ordinary* Voronoi diagram.

2.2 Multiplicatively Weighted Voronoi Diagram

Let $S = \{\mathrm{P}_1, \mathrm{P}_2, \cdots, \mathrm{P}_n\}$ be the set of points in the plane, and v_i be a positive real assigned to P_i for $i = 1, 2, \cdots, n$. For any point P, we call $\|\mathrm{P} - \mathrm{P}_i\|/v_i$ the *multiplicatively weighted distance*, and call v_i the *weight* assigned to P_i. We define region $R_{\mathrm{m}}(S; \mathrm{P}_i)$ by

$$R_{\mathrm{m}}(S; \mathrm{P}_i) = \{\mathrm{P} \mid \|\mathrm{P} - \mathrm{P}_i\|/v_i < \|\mathrm{P} - \mathrm{P}_j\|/v_j,\ j \neq i\}, \tag{2}$$

that is, $R_{\mathrm{m}}(S; \mathrm{P}_i)$ denotes the set of points that is closer to P_i than to any other P_j in terms of the multiplicatively weighted distance. The plane is partitioned into $R_{\mathrm{m}}(S; \mathrm{P}_1), R_{\mathrm{m}}(S; \mathrm{P}_2), \cdots, R_{\mathrm{m}}(S; \mathrm{P}_n)$. This partition is called the *multiplicatively weighted Voronoi diagram* [2,5].

A boundary of two Voronoi regions is a part of a circle, which is known as the Apollonius circle [6].

Fig. 1 shows an example of a multiplicatively weighted Voronoi diagram; the numbers in the parentheses represent the weights of the generators.

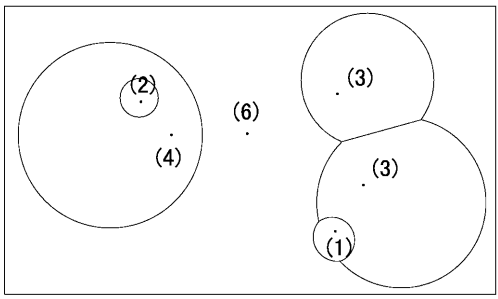


Fig. 1. Multiplicatively weighted Voronoi diagram

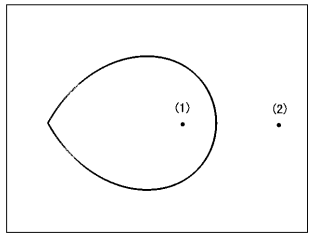


Fig. 2. Multiplicatively weighted crystal Voronoi diagram

2.3 Multiplicatively Weighted Crystal-Growth Voronoi Diagram

As in previous subsections, let $S = \{\mathrm{P}_1, \mathrm{P}_2, \cdots, \mathrm{P}_n\}$ be the set of generators in the plane and v_i be the weight assigned to P_i. Suppose that for each i, the i-th crystal grows from P_i by its own speed v_i. The crystals can grow only in empty areas; they cannot intrude into those areas that are already occupied by other crystals. Hence, a faster crystal must go around slower crystals. Thus, unlike the multiplicatively weighted distance, the time required for the i-th crystal to reach P is not determined by P and P_i only; it depends also on the locations and speeds of other crystals.

In this sense, the resulting crystal pattern is different from the multiplicatively weighted Voronoi diagram. This crystal pattern is called the *multiplicatively weighted crystal-growth Voronoi diagram*, or *the crystal Voronoi diagram* for short.

In the crystal Voronoi diagram, each crystal behaves as an obstacle against other crystals. Hence, for a point P in the i-th crystal region the distance from P_i to P should be measured along the shortest path completely included in the crystal.

Fig. 2 shows the crystal Voronoi diagram for two generators, P_1, P_2, with weights 1 and 2.

If all the growth speed v_i are the same, the crystal Voronoi diagram coincides with the ordinary Voronoi diagram.

Note that, unlike the multiplicatively weighted Voronoi diagram, the Voronoi region of a crystal Voronoi diagram is always connected. This is because a crystal cannot go through other crystals in the process of growing.

3 Simulation of the Crystal Growth

We can obtain the boundary for two crystals in the analytic form. But for three or more crystals, the calculation becomes difficult and complicated. In this section

we consider a method for computing the boundary curves approximately. For this purpose we employ the fast marching method for solving a certain type of a partial differential equation.

3.1 Fast Marching Method

Eikonal Equation. Let $\Omega \subset \mathbf{R}^2$ be a bounded region in the plane, and Γ be its boundary. Let $F(\boldsymbol{x})$ be a real-valued functions satisfying $F(\boldsymbol{x}) > 0$ for any $\boldsymbol{x} \in \Omega$. Furthermore, let $g(\boldsymbol{x})$ be a function on Γ. We consider a nonlinear partial differential equation

$$|\nabla u(\boldsymbol{x})| = F(\boldsymbol{x}) \quad \text{in } \Omega \tag{3}$$

with a boundary condition

$$u(\boldsymbol{x}) = g(\boldsymbol{x}) \quad \text{on } \Gamma, \tag{4}$$

where $F(\boldsymbol{x})$ and $g(\boldsymbol{x})$ are known and $u(\boldsymbol{x})$ is unknown. The equations (3) is called the *Eikonal equation*.

Assume that $1/F(\boldsymbol{x})$ represents the speed of a moving object at point $\boldsymbol{x}$ in Ω, and that $g(\boldsymbol{x}) = 0$ on Γ. Then, the solution $u(\boldsymbol{x})$ of the above Eikonal equation can be interpreted as the shortest time required for the object initially on the boundary Γ to reach the point $\boldsymbol{x}$. Therefore, we can use this equation to represent the behavior of the growth of a crystal. In particular, if $F(\boldsymbol{x}) = \infty$ in some area, this area behaves as an obstacle because the speed (i.e., $1/F(\boldsymbol{x})$) in this area is considered 0. This property is suitable to our purpose, because the areas occupied by crystals behave as obstacles to other crystals. In what follows, we assume that $g(\boldsymbol{x}) = 0$ on Γ.

To solve the equation (3) together with the boundary condition (4), Sethian [4] proposed a finite-difference method, called the *fast marching method*. In the finite-difference method, the unknown continuous function $u(\boldsymbol{x}) = u(x, y)$ is replaced by a finite set of values at discretized points

$$u_{i,j} = u(i\Delta x, j\Delta y), \tag{5}$$

where Δx and Δy are small values representing the interval for discretization in the x and y directions. We set the values of $u_{i,j}$'s on Γ being 0, and starting with these boundary points, we compute the values of the other $u_{i,j}$'s in the increasing order.

Apparently similar techniques have already been used in digital picture processing; they are called *distance-transformation* methods [7]. But usually the obtained distance is either L_1-distance or L_∞-distance, which is different from what we want to obtain, i.e., the Euclidean distance. Algorithms for obtaining the Euclidean distance are also proposed in digital image processing [8,9], but they cannot treat the obstacles, and hence cannot be applied to our purpose.

Finite-Difference Equation in the First Marching Method. Using the discretized value $u_{i,j}$, Sethian proposed finite-difference approximations of the equation (3). The most basic approximation is the first-order finite-difference equation defined by

$$[\max(D_{i,j}^{-x}u, -D_{i,j}^{+x}u, 0)^2 + \max(D_{i,j}^{-y}u, -D_{i,j}^{+y}u, 0)^2]^{1/2} = F_{i,j}, \tag{6}$$

where

$$\begin{aligned} D_{i,j}^{-x}u &= \frac{u_{i,j} - u_{i-1,j}}{\Delta x}\ ,\ D_{i,j}^{+x}u = \frac{u_{i+1,j} - u_{i,j}}{\Delta x}, \\ D_{i,j}^{-y}u &= \frac{u_{i,j} - u_{i,j-1}}{\Delta y}\ ,\ D_{i,j}^{+y}u = \frac{u_{i,j+1} - u_{i,j}}{\Delta y}, \\ F_{i,j} &= F(i\Delta x, j\Delta y). \end{aligned} \tag{7}$$

Eq. (6) is used to compute the unknown value $u_{i,j}$ from given u values at the upwind neighbor points and given $F_{i,j}$[4].

Sethian also proposed the second-order approximation of eq. (3) by

$$\left(\begin{array}{c} \max[[D_{i,j}^{-x}u + switch_{i,j}^{-x}\frac{\Delta x}{2}(D_{i,j}^{-x})^2u], \\ -[D_{i,j}^{+x}u + switch_{i,j}^{+x}\frac{\Delta x}{2}(D_{i,j}^{+x})^2u], 0]^2 \\ + \\ \max[[D_{i,j}^{-y}u + switch_{i,j}^{-y}\frac{\Delta y}{2}(D_{i,j}^{-y})^2u], \\ -[D_{i,j}^{+y}u + switch_{i,j}^{+y}\frac{\Delta y}{2}(D_{i,j}^{+y})^2u], 0]^2 \end{array} \right)^{\frac{1}{2}} = F_{i,j}, \tag{8}$$

where

$$switch_{i,j}^{\pm x} = \begin{cases} 1, & \text{if } u_{i\pm2,j} \text{ and } u_{i\pm1,j} \text{ are known and } u_{i\pm2,j} \le u_{i\pm1,j}, \\ 0, & \text{otherwise} \end{cases} \tag{9}$$

and $switch_{i,j}^{\pm y}$ is defined similarly.

The coefficient *switch* in eq. (8) is necessary, because $F(\boldsymbol{x})$ depends on $\boldsymbol{x}$ so that the shortest path might be curved, and consequently $u_{i-2,j}$, for example, might not be known even if the upwind-neighbor value $u_{i-1,j}$ is known.

For our purpose of computing the crystal Voronoi diagram, we use the first-order approximations to choose the upwind neighbors, and use the second-order approximation to compute the value of $u_{i,j}$.

Original Fast Marching Algorithm. The original fast marching algorithm proposed by Sethian is as follows.

Algorithm 1 (Fast marching method)

Step 1 (Initialization). Cover the region Ω with grid points $(i\Delta x, j\Delta y)$. Initialize KNOWN to be the set of all grid points on the boundary Γ, and TRIAL to be the set of all points that are one-grid far from KNOWN, and FAR to be the set of all the other points. Initialize the value $u_{i,j}$ as $u_{i,j} = 0$ for points in KNOWN, $u_{i,j} = \inf$ for points in FAR, and determine the value of $u_{i,j}$ according to eq. (8) for points in TRIAL.

Step 2 (Main loop). Repeat Steps 2.1 to 2.5.

2.1. From TRIAL choose and delete the point, say Q, with the smallest u value, and add it to KNOWN.

2.2. For each of the four neighbors of Q that is in FAR, move it from FAR to TRIAL.

2.3. For each of the four neighbors of Q that are in TRIAL, compute the u value using eq. (8). (If the point already has the u value, update it only if the new u value is smaller than the old one.)

2.4. If TRIAL is empty, stop. Otherwise go to 2.1. ∎

If we use a heap for representing and manipulating the set TRIAL, this algorithm runs in $O(N \log N)$ time for N grid points. Refer to [4] for the details of this algorithm.

3.2 Computation of the Crystal Voronoi Diagram

We apply the fast marching method to the simulation of the growth of crystals. We discretize the region in which we want to compute the crystal structure into grid points, and assign the generators to the nearest grid points, say $P_1, P_2, \cdots, P_n$. Let N be the total number of the grid points. We assign sequential numbers to all the grid points, and name them as $Q_1, Q_2, \cdots, Q_N$. Basically we follow Algorithm 1, but in several points we change it in the following way.

First, for each grid point Q_j, we assign the "crystal name" CNAME$[Q_j]$, which represents the ordinal number of the crystal to which Q_j belongs. The value of CNAME$[Q_j]$ is either an integer from 1 to n or "NONE". At the initial stage, we set CNAME$[P_k] = k$ for all the generators P_k, $k = 1, 2, \cdots, n$, set CNAME$[Q_j] = k$ for grid point Q_j that is one-grid far from P_k, and set CNAME$[Q_j]$ =NONE for the other grid points. Whenever the k-th crystal reaches Q_j, CNAME$[Q_j]$ is changed to k.

Secondly, at the initial stage, we set KNOWN to be the set $\{P_1, P_2, \cdots, P_n\}$ of the generators.

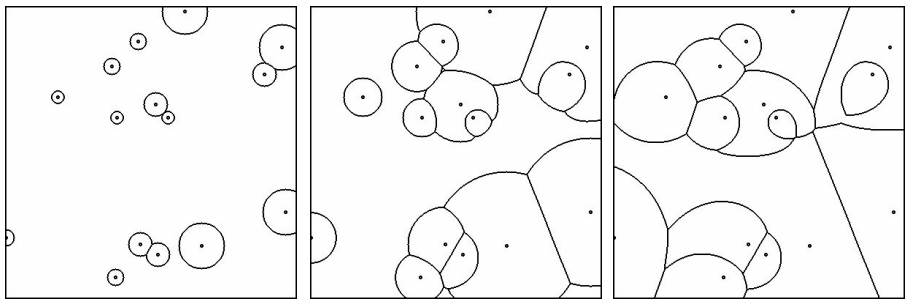
Thirdly, for the computation of the u value of a four-neighbor point, say Q_j, in TRIAL of the point Q in Step 1 or in Step 2.3 in Algorithm 1, we slightly modify the procedure in the following way.

(i) We read the crystal name k =CNAME[Q], and use the growth speed of the k-th crystal, that is, we substitute $F_{i,j} = 1/v_k$ to eq. (8).

(ii) We use the u values of only those points Q_l that are included in the k-th crystal, i.e., CNAME$[Q_l] = k$, in solving eq. (8).

(iii) Because of the above modifications (i) and (ii), the resulting u value is not necessary smaller than the previous value. Hence, only when the recomputed u value is smaller than the present value, we update the u value, and change CNAME$[Q_j]$ to k.

The output of the fast marching method modified as described above can be interpreted as the crystal Voronoi diagram in the sense that each grid point Q_j belongs to the crystal CNAME$[Q_j]$.



(a) t = 30 (b) t = 100 (c) Crystal Voronoi diagram

Fig. 3. Simulation of crystal Voronoi diagram by the fast marching method (t means the radius of the fastest growing crystal when the width between grids is one)

Fig. 3 shows the behavior of the algorithm. Here, the square region was replaced by 400×400 grid points and 15 generators were placed. Fig. 3 (a) and (b) show the frontiers of the crystals at the stage where the fastest crystal grows 30 times the grid distance and 100 times the grid distance, respectively. Fig. 3 (c) shows the final result.

4 Application to Path Planning

4.1 Fast Marching Method for Collision-Free Path

Sethian applied the fast marching method to the collision-free path among static obstacles [4]. Here, we extend his idea, and propose a method for finding a collision-free path among moving competitive robots. First, let us review Sethian's idea [4].

The Eikonal equation (3) can be written in the integral form as

$$u(\boldsymbol{x}) = \min_{\gamma} \int_{A}^{\boldsymbol{x}} F(\gamma(\tau))\mathrm{d}\tau, \tag{10}$$

where A is a start point, γ is a path from A to $\boldsymbol{x}$ in Ω. Thus, $u(\boldsymbol{x})$ represents the shortest time in which a robot can move from A to $\boldsymbol{x}$. Suppose that we get $u(\boldsymbol{x})$ for every point $\boldsymbol{x}$ in Ω using the fast marching method. Next, for any point B in Ω, the solution $X(t)$ of equation

$$X(t) = -\nabla u, \quad X(0) = B \tag{11}$$

gives the shortest path from A to B.

This idea can be extended to the case where the robot has its own shape instead of just a point. Suppose, for example, that a moving robot is a rectangle. Let (x, y) be the location of the center of the robot and θ be the angle of the longer edge of the rectangle with respect to the positive x direction; we measure

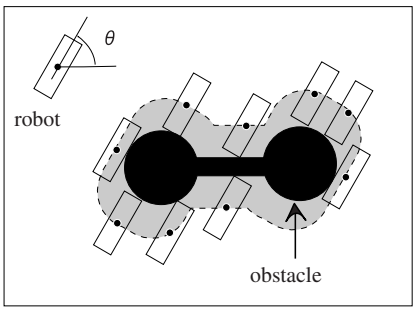


Fig. 4. The area where the robot's center cannot enter when it rotates at an angle of θ

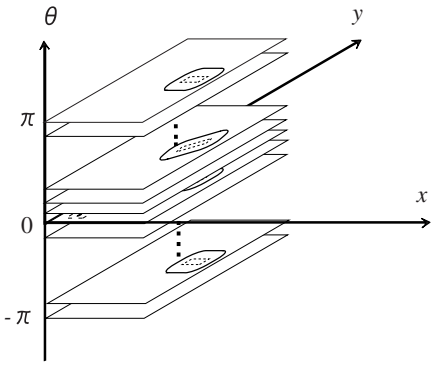


Fig. 5. 3-dimensional space of fast marching method for robot navigation

the angle counterclockwise. Thus the position and the posture of the robot can be represented by a point (x, y, θ) in a three-dimensional parameter space.

Next for each θ, we find the region in which the robot cannot enter without colliding the obstacle, as shown by the shaded area in Fig. 4. The boundary of this region can be obtained as the trajectory of the center of the robot that moves around keeping in contact with the obstacle. For this fixed θ, to consider the rectangular robot moving around the original obstacle is equivalent to consider a point robot moving around the extended region. Thus, we can reduce the problem of the moving robot among the obstacles to the problem of a moving point among the enlarged obstacles.

However, this reduction should be done for each value of θ. Hence, we discretize θ as well as x and y, and construct the three-dimensional grid structure as shown in Fig. 5. A fixed value of θ corresponds to a horizontal plane, in which we extend the obstacles.

Sethian used the fast marching method to solve the Eikonal equation

$$\left[\left(\frac{\partial u}{\partial x}\right)^2 + \left(\frac{\partial u}{\partial y}\right)^2 + \alpha\left(\frac{\partial u}{\partial \theta}\right)^2\right]^{1/2} = 1 \tag{12}$$

in the three-dimensional (x, y, θ) space. The partial derivatives $\partial u/\partial x$ and $\partial u/\partial y$ represent the inverses of x and y components of the velocity while $\partial u/\partial \theta$ represents the inverse of the angular velocity. The coefficient α represents the ratio of the time to translate the robot by unit length over the time to rotate the robot by unit angle.

4.2 Extension to Competitive Robots

Here we consider the situation where our robot moves among enemy robots. Suppose that our robot has an arbitrary shape while the enemy robots are circles,

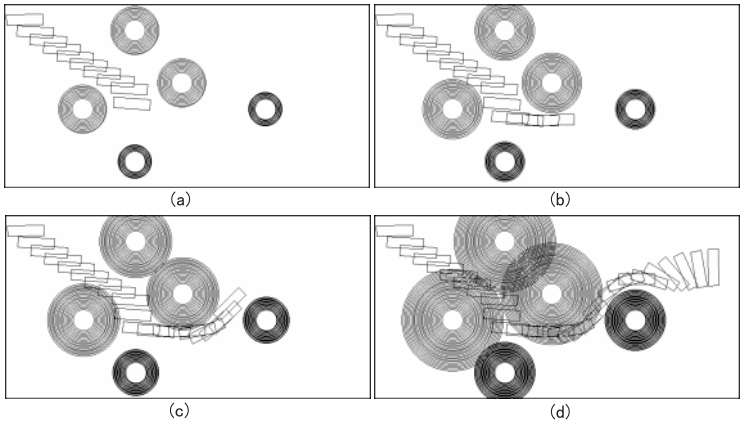


Fig. 6. Optimal answers of the robot navigation problems.

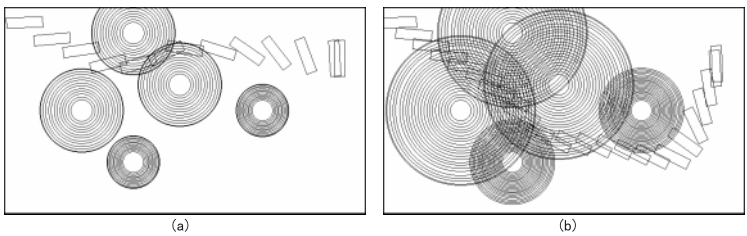


Fig. 7. Optimal answers of the robot navigation problems for other robot velocities

and each robot has its own velocity. Our robot wants to move avoiding enemies from the start point to the goal as fast as possible, while the enemy robots try to attack it. In this situation we want to find the worst-case optimal path from the start point to the goal.

For this purpose, we can apply the first marching method. The only difference from Sethian's path planning is that the obstacles are not static; they move with the intention to attack our robot. Hence, as we extended Sethian's fast marching method to the crystals, we treat the enemy robots as if they are crystals growing isotropically in every direction; these crystal regions represent the maximum area that the enemy robot can reach.

Fig. 6 shows an example of the collision-free path found by our method. The five enemy robots, starting with the initial circles representing the sizes of the robots, grow their regions by their own speed. Our robot, on the other hand, is a rectangle that can translate and rotate. In Fig. 6, (a), (b) and (c) show the status at some instants, while (d) shows the whole path of the robot to reach the goal. Fig. 7 (a) shows the generated path for the case that our robot can move faster than in Fig. 6, while Fig. 7 (b) shows the case that our robot moves more slowly than in Fig. 6.

5 Concluding Remarks

This paper studied the crystal Voronoi diagram from the computational point of view. First, we presented a method for computing the approximated diagram, where we modified the fast marching method to solve the Eikonal equation.

The approximation method proposed by Schaudt and Drysdale [1] requires $O(n^3)$ time for n crystals, whereas our new method runs in $O(N \log N)$ time for N grid points. This time complexity does not depend on the number of crystals.

Furthermore, we applied the crystal Voronoi diagram to the collision-free path planning among enemy robots, and evaluated our method by computational experiments.

One of the main problems for future is to raise the efficiency of the method. We might decrease the computational cost by using a course grid together with interpolation techniques. We might also decrease the memory cost by discarding the u values except around the frontiers of the crystals.

In our application to the path planning among competitive robots, we assumed that the enemy robots are circles. To generalize our method for arbitrary enemy shapes is another important problem for future.

Acknowledgements. The authors express their thanks to Prof. K. Hayami, Mr. T. Nishida and Mr. S. Horiuchi of the University of Tokyo for valuable comments. This work is supported by Toray Science Foundation, and the Grant-in-Aid for Scientific Research of the Japanese Ministry of Education, Science, Sports and Culture.

References

1. B.F. Schaudt and R.L. Drysdale: Multiplicatively weighted crystal growth Voronoi diagram. *Proceedings of the Seventh Annual Symposium on Computational Geometry* (North Conway, June 1991), pp. 214–223.
2. F. Aurenhammer: Voronoi diagrams—A survey of a fundamental Geometric data structure. *ACM Computing Surveys*, vol. 23, no. 3 (1991), pp. 345–405.
3. A. Okabe, B. Boots, and K. Sugihara: *Spatial Tessellations—Concepts and Applications of Voronoi Diagrams*. John Wiley, Chickester, 1992.
4. J.A. Sethian: Fast marching methods. *SIAM Review*, vol. 41, no. 2 (1999), pp. 199–235.
5. C.A. Wang and P.Y. Tsin: Finding constrained and weighted Voronoi diagrams in the plane. *Proceedings of the Second Canadian Conference in Computational Geometry* (Ottawa, August 1990), pp.200–203.
6. D. Pedoe: *Geometry—A Comprehensive Course*. Cambridge University Press, London, 1970.
7. A. Rosenfeld and J. Pfalts: Sequential operations in digital picture processing. *Journal of ACM*, vol. 13 (1966), pp. 471–494.
8. L. Chen and H.Y.H. Chuang. A fast algorithm for Euclidean distance maps of a 2-d binary image. *Infor. Process. Lett.* vol. 51 (1994), pp. 25–29.
9. T. Hirata: A unified linear-time algorithm for computing distance maps. *Infor. Process. Lett.*, vol. 58 (1996), pp. 129–133.

The Voronoi-Delaunay Approach for Modeling the Packing of Balls in a Cylindrical Container

V.A. Luchnikov[1], N.N. Medvedev[1], M.L. Gavrilova[2]
[1] Institute of Chemical Kinetics and Combustion, 630090 Novosibirsk, Russia
luchnik,nikmed@ns.kinetics.nsc.ru
[2] Dept of Comp. Science, University of Calgary, AB, Canada, T2N1N4
marina@cpsc.ucalgary.ca

Abstract. The paper presents an approach for calculation of the Voronoi network of a system of balls con ned inside a cylindrical container. We propose to consider a boundary of a container as one of the elements of the system. Then the Voronoi network can be built for a system containing non-spherical particles. An explicit formula to compute the coordinates of the Voronoi vertex between a three balls and a cylinder are obtained. The approach is implemented in 3D and tested on the models of balls packing with di erent structure.

1 Introduction

The Voronoi ideas, that are well known in mathematics and computer science, have been used extensively to solve many applied problems in physics, mechanics and chemistry [7, 6]. Originally, the Voronoi-Delaunay approach in physics was applied to study the structure of disordered packing of balls and models of liquid and glasses [3]. The method is also a helpful tool for analysis of voids: empty spaces between atoms, where the Voronoi network plays the role of a navigation map [5]. This property of the Voronoi network has been used in studying various problems, in particular to model the permeability and the fluid flow through the packing of balls [1, 10].

Traditionally, the Voronoi-Delaunay method is applied to models with the periodic boundary conditions (used to simulate an infinite media) or to models with an open boundary (such as biological molecules. However, in many physical-chemical problems the boundary plays the determinative role. For example, a typical chemical reactor is a cylinder filled with the spherical granules. To simulate flows through the packing in the cylinder one should create an algorithm that takes into account the boundary of the system. From the mathematical point of view, the problem is similar to building the medial axes inside a cylindrical cavity containing balls. However, the known approaches to compute the medial axes inside a cavity [8, 9] are complicated and thus deem to be inefficient for the analysis of models with a large number of balls.

In this paper, we present an efficient approach for the calculation of the Voronoi network of a packing of balls inside a cylinder. We propose to consider a cylindrical boundary as an additional non-spherical element of the system, and provide an explicit formula to compute the coordinates of the Voronoi vertex between the three balls and a cylindrical boundary. The algorithm was implemented and tested in 3D for packings with different structure.

V.N. Alexandrov et al. (Eds.): ICCS 2001, LNCS 2073, pp. 748–752, 2001.
©Springer-Verlag Berlin Heidelberg 2001

2 The algorithm

One of the possible ways to compute the Voronoi network for a set of balls in a cylinder is to use the algorithm presented in [4], where the Voronoi network was calculated for 3D systems of straight lines and sphere-cylinders. The algorithm is based on the idea of *the Delaunay empty sphere* [7]. Let us assume that the empty sphere moves inside the system so that it touches at least three objects at any moment of time. In this case the center of the sphere moves along an edge of the 3D Voronoi network. If the distance from any point in the space to any object is expressed by explicit functions d, then the trajectory of the center of the Delaunay empty sphere can be computed numerically by performing the series of small shifts along the edge. The direction of the shift $\mathbf{v}$ is found from the equation: $(\nabla d_i \cdot \mathbf{v})|_{\mathbf{r}} = (\nabla d_j \cdot \mathbf{v})|_{\mathbf{r}} = (\nabla d_k \cdot \mathbf{v})|_{\mathbf{r}}$, where the indices i, j, k enumerate the objects touched by the sphere. For a cylindrical wall we use the following distance function: $d_c = R_c - (x^2 + y^2)^{1/2}$, where R_c is the radius of the cylinder, x and y are the coordinates of a point inside the cylinder, provided the origin is on the axis of the cylinder. d_c is a differentiable function of the coordinates, except for the axis of the cylinder. The advantage of this method is in its simplicity and versatility: it can be used to build the Voronoi network for the system of any convex non-spherical objects for which there is no explicit formula to compute the coordinates of the Voronoi vertex. However this approach is rather time consuming: as it was shown on a packing of balls of different radii this algorithm runs 20 times slower then the algorithm based on explicit calculation of the coordinates of the Voronoi vertex [4]. For the problem of balls in cylinder, we can find a formula for the Voronoi vertex explicitly.

3 The Empty Sphere Problem

The task is to inscribe a sphere between the cylinder and three balls, that are located inside the cylinder and do not intersect. For simplicity, assume the cylinder is vertical. Denote the sphere with the smallest radius as (x_4, y_4, z_4, r_4). Choose the origin at the center of this sphere. Now, we apply the technique similar to one presented in [2]. We shrink all the balls by r_4, increase cylinder radius by r_4, and obtain the set of equations representing the condition that the inscribed sphere with the center (x, y, z) and the radius r touches three spheres and the cylinder:

$$\begin{cases} (x - x_1)^2 + (y - y_1)^2 = (r_1 - r)^2 \\ (x - x_2)^2 + (y - y_2)^2 + (z - z_2)^2 = (r + r_2)^2 \\ (x - x_3)^2 + (y - y_3)^2 + (z - z_3)^2 = (r + r_3)^2 \\ x^2 + y^2 + z^2 = r^2. \end{cases}$$

Here (x_1, y_1, r_1) are the coordinates of the axis of the cylinder and its radius, and (x_i, y_i, z_i, r_i), $i = 2, 3$ are the coordinates and radii of the two remaining

spheres. Subtracting the last equation from the first three we arrive to:

$$\mathbf{A}\begin{bmatrix} x \\ y \\ r \end{bmatrix} = \mathbf{b}, \mathbf{A} = \begin{bmatrix} x_1 & y_1 & -r_1 \\ x_2 & y_2 & r_2 \\ x_3 & y_3 & r_3 \end{bmatrix}, \mathbf{b} = \begin{bmatrix} l_1 - z^2 \\ l_2 - 2zz_2 \\ l_3 - 2zz_3 \end{bmatrix},$$

where $l_1 = {x_1}^2 + {y_1}^2 - {r_1}^2$, $l_i = {x_i}^2 + {y_i}^2 + {z_i}^2 - {r_i}^2$, $i = 2, 3$.
Solving the above system, we obtain:

$$x = \frac{1}{2|\mathbf{A}|}\begin{vmatrix} l_1 - z^2 & y_1 & -r_1 \\ l_2 - 2zz_2 & y_2 & r_2 \\ l_3 - 2zz_3 & y_3 & r_3 \end{vmatrix}, y = \frac{1}{2|\mathbf{A}|}\begin{vmatrix} x_1 & l_1 - z^2 & -r_1 \\ x_2 & l_2 - 2zz_2 & r_2 \\ x_3 & l_3 - 2zz_3 & r_3 \end{vmatrix},$$

$$r = \frac{1}{2|\mathbf{A}|}\begin{vmatrix} x_1 & y_1 & l_1 - z^2 \\ x_2 & y_2 & l_2 - 2zz_2 \\ x_3 & y_3 & l_3 - 2zz_3 \end{vmatrix},$$

where $|\mathbf{A}|$ denotes the determinant of the matrix. When $|\mathbf{A}| = 0$, we arrive to the degenerate case, i.e. there are infinitely many inscribed spheres. Thus, let us assume that $|\mathbf{A}| \neq 0$. The obtained expressions are substituted into the last equation $x^2 + y^2 + z^2 - r^2 = 0$, which yields the 4th degree polynomial of z:

$$az^4 + bz^3 + cz^2 + dz + e = 0$$

The coefficients in the above equation are in the form: $a = {A_x}^2 + {A_y}^2 - {A_r}^2$, $b = 2(A_xB_x + A_yB_y - A_rB_r)$, $c = {B_x}^2 + 2A_xC_x + {B_y}^2 + 2A_yC_y + 4|\mathbf{A}|^2 - {B_r}^2 - 2A_rC_r$, $d = 2(B_xC_x + B_yC_y - B_rC_r)$, $e = {C_x}^2 + {C_y}^2 - {C_r}^2$, where

$$A_x = \begin{vmatrix} -1 & y_1 & -r_1 \\ 0 & y_2 & r_2 \\ 0 & y_3 & r_3 \end{vmatrix}, B_x = -2\begin{vmatrix} 0 & y_1 & -r_1 \\ z_2 & y_2 & r_2 \\ z_3 & y_3 & r_3 \end{vmatrix}, C_x = \begin{vmatrix} l_1 & y_1 & -r_1 \\ l_2 & y_2 & r_2 \\ l_3 & y_3 & r_3 \end{vmatrix},$$

$$A_y = \begin{vmatrix} x_1 & -1 & -r_1 \\ x_2 & 0 & r_2 \\ x_3 & 0 & r_3 \end{vmatrix}, B_y = -2\begin{vmatrix} x_1 & 0 & -r_1 \\ x_2 & z_2 & r_2 \\ x_3 & z_3 & r_3 \end{vmatrix}, C_y = \begin{vmatrix} x_1 & l_1 & -r_1 \\ x_2 & l_2 & r_2 \\ x_3 & l_3 & r_3 \end{vmatrix},$$

$$A_r = \begin{vmatrix} x_1 & y_1 & -1 \\ x_2 & y_2 & 0 \\ x_3 & y_3 & 0 \end{vmatrix}, B_r = -2\begin{vmatrix} x_1 & y_1 & 0 \\ x_2 & y_2 & z_2 \\ x_3 & y_3 & z_3 \end{vmatrix}, C_r = \begin{vmatrix} x_1 & l_1 & l_1 \\ x_2 & l_2 & l_2 \\ x_3 & l_3 & l_3 \end{vmatrix}.$$

The equation is then solved for z. The final answer is:

$$x_f = \frac{(A_xz^2 + B_xz + C_x)}{2|\mathbf{A}|} + x_4, y_f = \frac{(A_yz^2 + B_yz + C_y)}{2|\mathbf{A}|} + y_4,$$

$$r_f = \frac{(A_rz^2 + B_rz + C_r)}{2|\mathbf{A}|} - r_4, z_f = z + z_4.$$

Up to four solutions are possible. However, the solutions with imaginary and negative r are non-physical and omitted.

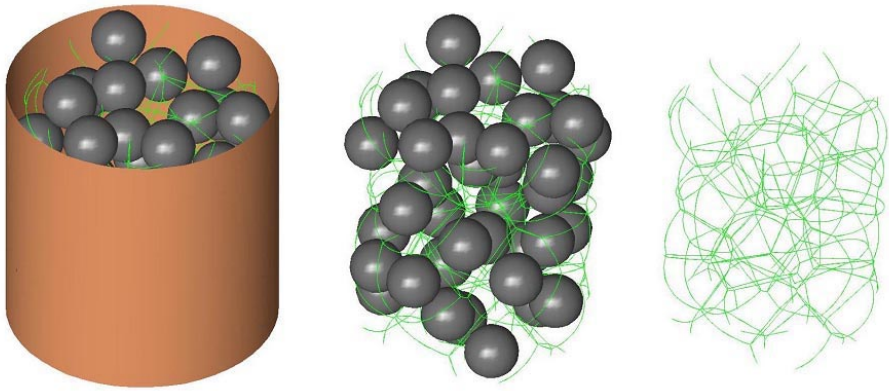


Figure 1: A cylinder with the balls (a), the arrangement of the balls in the cylinder (b), the Voronoi network (edges of the Voronoi diagram) taking into consideration the cylinder (c).

The algorithm for Voronoi network calculation was implemented in Fortran. Fig. 1 illustrate its application for a system of 40 balls with equal radii $r = 0.2$ in a cylinder of radii $r_c = 2.0$. One can note that the Voronoi edges that are at the center of the cylinder are segments of straight lines. They are edges of the standard Euclidean Voronoi diagram. While the edges at the cylinder surface are curved. This situation is typical for systems with the non-spherical particles (the edges are on the intersection of curved quadratic surfaces). The algorithm was also tested on a system representing dense packing of 300 Lennard-Jones atoms with different structure. Two models: one is a model with disordered packing obtained by Monte-Carlo relaxation in a cylinder with a fixed value of the diameter $D = 6\sigma$ (where σ is the parameter of the Lennard-Jones potential) and the other model with the crystalline-like structure (made by slightly varying the diameter of cylinder). The reslts show that the largest channels (Voronoi network bonds with the largest bottle-necks) occur near to the wall of the cylinder. This is an anticipated result, since a flow of liquid through a packing of balls is inhomogeneous and the main streams are at the cylinder wall. It was also noted that a fraction of large channels along the wall is higher for the model with the disordered packing than for the crystalline-like model.

4 Future Work and Acknowledgments

One of the possible directions of the future research is extension of the method to handle different types of curvlinear boundaries and the experimentation with

physical systems built inside given boundaries. The work was supported in part by SB RAS No.46, RFFI No.01-03-32903 and UCRS grants. We also would like to thank Dr. Annie Gervois for helpful comments and suggestions.

References

[1] Bryant, S. and Blunt, M. , Phys. Rev. A, **46(4)** (1992) 2004

[2] Gavrilova, M. and Rokne, J. Swap conditions for dynamic Voronoi diagram for circles and line segments, Comp-Aided Geom. Design, **16** (1999) 89–106

[3] Finney, J. Random packings and the structure of simple liquids. Roy.Soc.London, **319** (1970) 479–495

[4] Luchnikov, V.A., Medvedev,N.N., Oger, L. and Troadec, J.-P. The Voronoi-Delaunay analysis of voids in system of nonspherical particles. Phys.Rev.E. **59(6)**, (1999) 7205–7212

[5] Medvedev, N.N. Computational porosimetry, in *Voronoi's impact on modern science*. Ed. P. Engel, H. Syta, Inst. of Math., Kiev., (1998) 164–175

[6] Medvedev, N.N. Voronoi-Delaunay method for non-crystalline structures, SB Russian Academy of Science, Novosibirsk, (2000)

[7] Okabe, A., Boots, B., Sugihara, K. Spatial tesselation concepts and applications of Voronoi diagrams, J. Wiley & Sons, Chichester, England (1992)

[8] Rowe, N.C. Obtaining Optimal Mobile-Robot Paths with Non-Smooth Anisotropic Cost Functions, J. Robot. Res, **16(3)** (1997) 375–399

[9] Sherbrooke E. C., Patrikalakis N.M. and Brisson, N. An Algorithm for the Medial Axis Transform of 3D Polyhedral Solids, IEEE Trans. Visualiz. Comp. Graph, **2(1)** (1996) 45–61

[10] Thompson, K.E and Fogler, H.S. Modelling flow in disordered packed bed from pore-scale fluid mechanics. AICHE Journal, **43(6)** (1997) 1377–1389

Multiply Guarded Guards in Orthogonal Art Galleries

T.S. Michael[1] and Val Pinciu[2]

[1] Mathematics Department, United States Naval Academy
Annapolis, MD 21402
tsm@nadn.navy.mil
[2] Mathematics Department, Southern Connecticut State University
New Haven, CT 06515
pinciu@scsu.ctstateu.edu

Abstract. We prove a new theorem for orthogonal art galleries in which the guards must guard one another in addition to guarding the polygonal gallery. A set of points $\mathcal{G}$ in a polygon P_n is a k-guarded guard set for P_n provided that (i) for every point x in P_n there exists a point w in $\mathcal{G}$ such that x is visible from w ; and (ii) every point in $\mathcal{G}$ is visible from at least k other points in $\mathcal{G}$. The polygon P_n is orthogonal provided each interior angle is 90° or 270°. We prove that for $k \geq 1$ and $n \geq 6$ every orthogonal polygon with n sides has a k-guarded guard set of cardinality $k\lfloor n/6 \rfloor + \lfloor (n+2)/6 \rfloor$; this bound is best possible. This result extends our recent theorem that treats the case $k = 1$.

1 Introduction

Throughout this paper P_n denotes a simple closed polygon with n sides, together with its interior. A point x in P_n is *visible* from point w provided the line segment wx does not intersect the exterior of P_n. (Every point in P_n is visible from itself.) The set of points $\mathcal{G}$ is a *guard set* for P_n provided that for every point x in P_n there exists a point w in $\mathcal{G}$ such that x is visible from w. Let $g(P_n)$ denote the minimum cardinality of a guard set for P_n.

A guard set for P_n gives the positions of stationary guards who can watch over an art gallery with shape P_n, and $g(P_n)$ is the minimum number of guards needed to prevent theft from the gallery. Chvátal's celebrated Art Gallery Theorem [1] asserts that among all polygons with n sides ($n \geq 3$), the maximum value of $g(P_n)$ is $\lfloor n/3 \rfloor$.

Over the years numerous "art gallery problems" have been proposed and studied, in which different restrictions are placed on the shape of the galleries or the powers and responsibilities of the guards. (See the monograph by O'Rourke [7] and the survey by Shermer [8].) For instance, in an *orthogonal* polygon P_n each interior angle is 90° or 270°, and thus the sides occur in two perpendicular orientations, say, horizontal and vertical. An orthogonal polygon must have an even number of sides. For even $n \geq 4$ we define

$$g_\perp(n) = \max\{g(P_n) : \ P_n \text{ is an orthogonal polygon with } n \text{ sides}\}.$$

V.N. Alexandrov et al. (Eds.): ICCS 2001, LNCS 2073, pp. 753–762, 2001.
© Springer-Verlag Berlin Heidelberg 2001

Kahn, Klawe, and Kleitman [3] gave a formula for $g_\perp(n)$:

Orthogonal Art Gallery Theorem *For $n \geq 4$ we have $g_\perp(n) = \lfloor n/4 \rfloor$.*

A set of points $\mathcal{G}$ in a polygon P_n is a *k-guarded guard* set for P_n provided that

(i) for every point x in P_n there exists a point w in $\mathcal{G}$ such that x is visible from w, i.e., $\mathcal{G}$ is a guard set for P_n; and
(ii) for every point w in $\mathcal{G}$ there are k points in $\mathcal{G}$ different from w from which w is visible.

In our art gallery scenario a k-guarded guard set prevents theft from the gallery *and* prevents the ambush of an insufficiently protected guard. We define the parameter

$$gg(P_n, k) = \min\{|\mathcal{G}| : \mathcal{G} \text{ is a } k\text{-guarded guard set for } P_n\}.$$

Liaw, Huang, and Lee [4], [5] refer to a 1-guarded guard set for a polygon P_n as a *weakly cooperative* guard set and show that the computation of $gg(P_n, 1)$ is an NP-hard problem. Let

$$gg_\perp(n, k) = \max\{gg(P_n, k) : \ P_n \text{ is an orthogonal polygon with } n \text{ sides}\}.$$

The authors [6] have recently determined the function $gg_\perp(n, 1)$.

Proposition 1. *For $n \geq 6$ we have $gg_\perp(n, 1) = \lfloor n/3 \rfloor$.*

In this paper we extend Proposition 1 to the "multiply guarded" situations with $k \geq 2$. Here is our main result.

Theorem 1. *For $k \geq 1$ and $n \geq 6$ we have*

$$gg_\perp(n, k) = k \left\lfloor \frac{n}{6} \right\rfloor + \left\lfloor \frac{n+2}{6} \right\rfloor. \tag{1}$$

When $k = 1$, the expression in (1) simplifies to $\lfloor n/3 \rfloor$ in accordance with Proposition 1.

If k is large, and we require that the guards be posted at vertices of the polygon P_n, then some vertex must contain more than one guard, that is, the k-guarded guard set is actually a multiset. In our proof of Theorem 1 it is convenient to first allow multiple guards at the same vertex (§5), and then show that the guards can always be moved to distinct points (§6).

2 A Construction

We begin our proof of Theorem 1 by constructing extremal polygons. Let P_n denote the orthogonal polygon of "waves" in Figure 1. The full polygon is used in case $n \equiv 0 \pmod 6$, while the broken lines indicate the boundaries of a partial

wave when $n \equiv 2, 4 \pmod 6$. Let $\mathcal{G}$ be a k-guarded guard set for P_n. Each complete wave of P_n uses six sides and forces $k+1$ distinct points in $\mathcal{G}$. Also, when $n \equiv 4 \pmod 6$, the partial wave forces one additional point. Thus $|\mathcal{G}| \geq (k+1)\lfloor n/6 \rfloor$ for $n \not\equiv 4 \pmod 6$, and $|\mathcal{G}| \geq (k+1)\lfloor n/6 \rfloor + 1$ for $n \equiv 4 \pmod 6$. It follows from some algebraic manipulation that $gg_{\perp}(k,n) \geq |\mathcal{G}| \geq k\lfloor n/6 \rfloor + \lfloor (n+2)/6 \rfloor$ for $n \geq 6$.

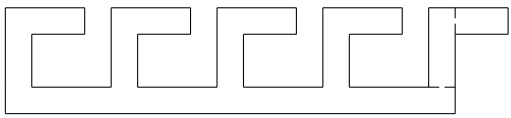


Fig. 1. Orthogonal polygon P_n for which $gg(P_n, k)$ is maximum

3 Galleries, Guards, and Graphs

Let P_n be a simple polygon with n sides. It is well known that diagonals may be inserted in the polygon P_n to produce a triangulation, that is, a decomposition of P_n into triangles. Diagonals may intersect only at their endpoints. The edge set in a *triangulation graph* T_n consists of pairs of consecutive vertices in P_n (the *boundary edges*) together with the pairs of vertices joined by diagonals (the *interior edges*) in a fixed triangulation. One readily shows that a triangulation graph is 3-colorable, that is, there exists a map from the vertex set to the color set $\{1, 2, 3\}$ such that adjacent vertices receive different colors.

Similarly, a *quadrangulation* Q_n of the polygon P_n is a decomposition of P_n into quadrilaterals by means of diagonals. We refer to Q_n as a *convex quadrangulation* provided each quadrilateral is convex. We also view Q_n as a *quadrangulation graph* in the expected manner. Note that Q_n is a plane bipartite graph with an even number of vertices. The (weak) planar dual of Q_n is a graph with a vertex for each bounded face of Q_n, where two vertices are adjacent provided the corresponding faces share an edge. The planar dual of a quadrangulation graph is a tree.

Let $G_n = (V, E)$ be a triangulation or quadrangulation graph on n vertices. We say that a set $\mathcal{G}$ of vertices is *guard set* of G_n provided every bounded face of G_n contains a vertex in $\mathcal{G}$. If, in addition, every vertex in $\mathcal{G}$ occurs in a bounded face with another vertex in $\mathcal{G}$, then $\mathcal{G}$ is a *guarded guard* set for G_n. We let $g(G_n)$ and $gg(G_n)$ denote the minimum cardinality of a guard set and guarded guard set, respectively, for the graph G_n.

4 The Proof of Proposition 1: Guarded Guards

Our proof of Theorem 1 relies on elements contained in our proof [6] of Proposition 1, which we review in this section. The strategy is to employ a coloring

argument in a triangulation graph as Fisk [2] did in his elegant proof Chvátal's Art Gallery Theorem. Our proof also depends on the following result, which was an important ingredient in the original proof [3] of the Orthogonal Art Gallery Theorem.

Proposition 2. *Every orthogonal polygon has a convex quadrangulation.*

The quadrangulation in Proposition 2 may always be selected so that each quadrilateral has positive area, (i.e., its four vertices do not fall on a line), and we shall always do so. However, quadrilaterals with three points on a line are sometimes unavoidable; these degenerate quadrilaterals are an issue in §6.

The proof of Proposition 1 relies on the following graph-theoretic result.

Proposition 3. *We have* $gg(Q_n) \leq \lfloor n/3 \rfloor$ *for each quadrangulation graph* Q_n *on* $n \geq 6$ *vertices.*

Proof Outline. The proof is illustrated in Figure 2. Let P_n be an orthogonal polygon with n sides, and let Q_n be the quadrangulation graph for the convex quadrangulation of P_n guaranteed by Proposition 2.

We construct a set $\mathcal{G}$ of vertices in Q_n that satisfies (i) $|\mathcal{G}| \leq \lfloor n/3 \rfloor$; (ii) every quadrilateral of Q_n contains a vertex of $\mathcal{G}$; (iii) every vertex in $\mathcal{G}$ is contained in a quadrilateral with another vertex in $\mathcal{G}$. Here is our strategy:

- We *triangulate* Q_n by inserting a diagonal in each bounded face to obtain a triangulation graph T_n with special properties.
- We 3-*color* the vertices of T_n. The least frequently used color gives us a set of vertices $\mathcal{G}'$ that satisfies conditions (i) and (ii).
- We *shift* some vertices of $\mathcal{G}'$ along edges of T_n to produce a set $\mathcal{G}$ that also satisfies condition (iii).

Triangulate: The graph Q_n and its planar dual are both bipartite, and hence we have the vertex biparition $V = V^+ \cup V^-$ and the face bipartition $F^+ \cup F^-$ as indicated in Figure 2(a). Each edge of Q_n joins a vertex in V^+ and a vertex in V^-. Each face f of Q_n contains two vertices in V^+ and two vertices in V^-. If $f \in F^+$, then we join the two vertices of f in V^+ by an edge, while if $f \in F^-$, we join the two vertices of f in V^- by an edge. The resulting graph is our triangulation T_n. (See Figure 2(b).) Let E_{diag} denote the set of edges added to Q_n by inserting a diagonal in each face in our triangulation process. Thus our triangulation graph is $T_n = (V, E \cup E_{\text{diag}})$.

3-*Color:* We 3-color the triangulation graph T_n. Let $\mathcal{G}'$ be the set of vertices of T_n in a color that occurs least frequently. Then $|\mathcal{G}'| \leq \lfloor n/3 \rfloor$; condition (ii) also holds. However, condition (iii) may fail, as in Figure 2(c).

Shift: Let Y denote the set of vertices in $\mathcal{G}'$ with degree 3 in T_n, and let X be the complement of Y in $\mathcal{G}'$. Then for each $y \in Y$ there is a unique "conjugate" vertex y^* such that $[y, y^*] \in E_{\text{diag}}$. Let $Y^* = \{y^* : \ y \in Y\}$ and define the set $\mathcal{G} = X \cup Y^*$.

In [6] we prove that the set $\mathcal{G}$ satisfies conditions (i)-(iii). Thus $\mathcal{G}$ is a guarded guard set for the quadrangulation graph Q_n, and $|\mathcal{G}| \leq \lfloor n/3 \rfloor$. □

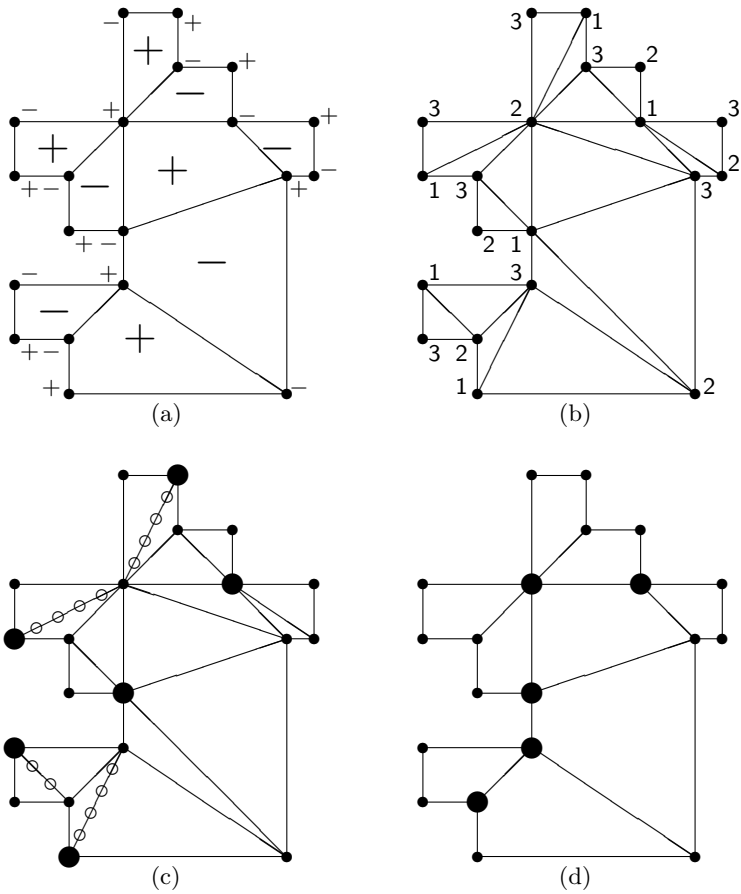


Fig. 2. The proof of Proposition 1 (a) The quadrangulation graph Q_n with vertex and face bipartitions indicated by + and − (b) The triangulation graph T_n and a 3-coloring (c) The guard set $\mathcal{G}'$; guards in $\mathcal{G}'$ at vertices of degree 3 are shifted along the indicated edges (d) The final guarded guard set $\mathcal{G}$ of Q_n

Now suppose that P_n is an orthogonal polygon. Then P_n has a convex quadrangulation Q_n by Proposition 2. The convexity of the quadrilateral faces implies that the guarded guard set $\mathcal{G}$ in Proposition 3 is a 1-guarded guard set for the orthogonal polygon P_n. Thus $gg(P_n, 1) \leq \lfloor n/3 \rfloor$. We constructed polygons to establish the reverse inequality in Figure 1. This completes the outline of our proof of Proposition 1. □

5 Proof of Theorem 1

Proposition 1 establishes Theorem 1 for $k = 1$. The proof for $k \geq 2$ is illustrated in Figure 3. Let P_n be an orthogonal polygon, and let Q_n be a convex quad-

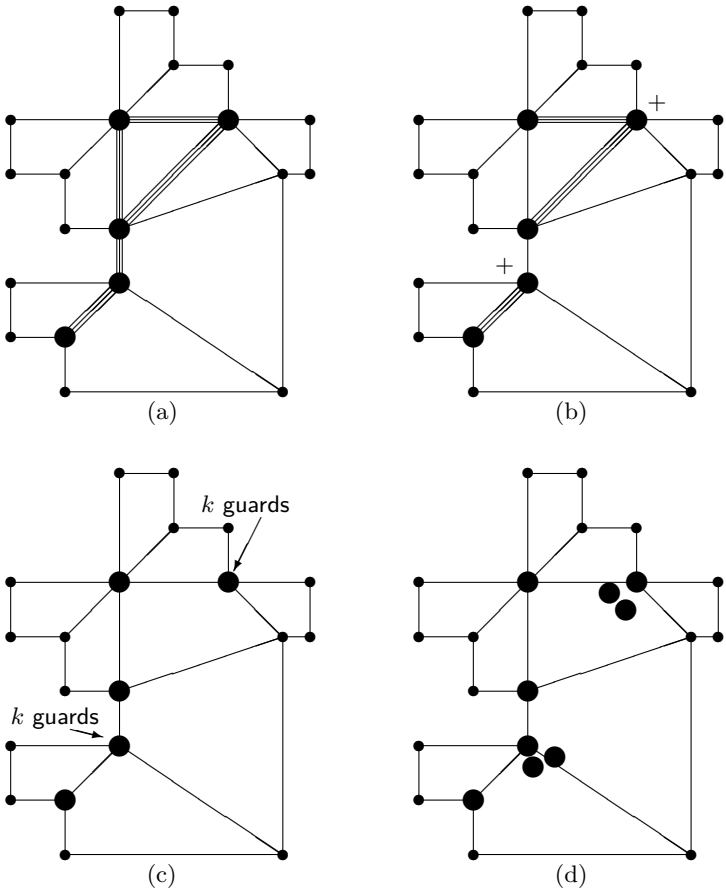


Fig. 3. The proof of Theorem 1 (a) The guarded guard set $\mathcal{G}$ of the quadrangulation graph Q_n from Figure 2 and the graph $G(\mathcal{G})$ (b) A spanning forest of stars $F(\mathcal{G})$ and a set of centers $\mathcal{G}^+$ (c) Selection of multiple guards at vertices in $\mathcal{G}^+$ (d) Separation of multiple guards for $k = 3$

rangulation of P_n. Let $\mathcal{G}$ denote the guarded guard set for the quadrangulation graph Q_n produced in the proof of Proposition 1. Now define a graph $G(\mathcal{G})$ whose vertex set is $\mathcal{G}$ with two vertices joined by an edge provided they are both contained in a quadrilateral face of Q_n. (See Figure 3(a).) No vertex of $G(\mathcal{G})$ is isolated because $\mathcal{G}$ is a guarded guard set of the graph Q_n. Therefore $G(\mathcal{G})$ has a spanning forest $F(\mathcal{G})$, where each component is a star. (See Figure 3(b).) Let $\mathcal{G}^+$ be the set of the centers of the stars. (Select either vertex as the center of a star with one edge.) Now $|\mathcal{G}^+| \leq \lfloor |\mathcal{G}|/2 \rfloor$. We insert $k-1$ additional guards at each vertex in $\mathcal{G}^+$ to obtain a *multiset* $\mathcal{G}^*$ of vertices of Q_n. Vertices may appear more than once in $\mathcal{G}^*$, but this is unavoidable if k is large and we require the guards to be placed at vertices of Q_n. Now each vertex of Q_n is visible from at

least k others. By Proposition 1 the cardinality of the multiset $\mathcal{G}^*$ satisfies

$$|\mathcal{G}^*| = |\mathcal{G}| + (k-1)|\mathcal{G}^+| \leq \left\lfloor \frac{n}{3} \right\rfloor + (k-1)\left\lfloor \frac{\lfloor n/3 \rfloor}{2} \right\rfloor = k\left\lfloor \frac{n}{6} \right\rfloor + \left\lfloor \frac{n+2}{6} \right\rfloor.$$

By the convexity of the quadrilateral faces of the orthogonal polygon P_n, each point in P_n is certainly visible from at least one guard, and so we have produced a k-guarded guard multiset $\mathcal{G}^*$ for P_n.

6 Separation of Guards and Degenerate Quadrilaterals

The k-guarded guard multiset $\mathcal{G}^*$ constructed in the previous section is satisfactory graph-theoretically, but not geometrically. With the same notation as in the previous section, we now prove that the k guards at each vertex w in $\mathcal{G}^+$ can always be separated to obtain a k-guarded guard *set* of points for P_n, as in Figure 3(d). This is a consequence of the following lemma.

Lemma 1. *Let Q_n be a convex quadrangulation of the orthogonal polygon P_n, and let w be a vertex of P_n. Then there exists a region R_w of points in P_n such that any vertex in the graph $G(\mathcal{G})$ adjacent to w is visible from every point in R_w.*

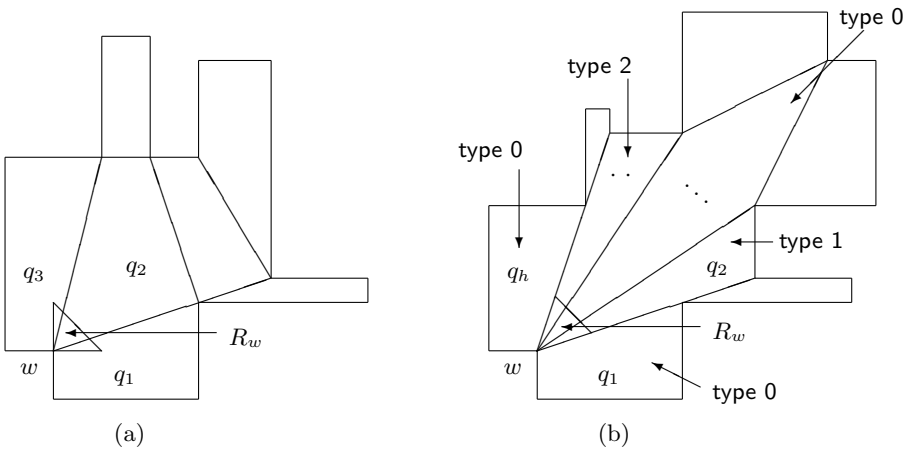


Fig. 4. The quadrilaterals $q_1, q_2, \ldots, q_h$ at vertex w are all visible from each point in a triangular region R_w for both (a) nondegenerate and (b) degenerate quadrilaterals

Proof. The main idea is depicted in Figure 4. If there are no degenerate quadrilaterals at w, then a small right triangular region in the "interior quadrant" at w serves as R_w. When degenerate quadrilaterals are present (with three points on a line), our proof is more complicated, and an acute triangular region serves as R_w.

If there is a 90° angle at w, one may readily show that there are no degenerate quadrilaterals at w. We now treat the case in which there is a 270° angle at w. Without loss of generality w is at the origin in the Cartesian plane, and P_n has edges along the negative x- and y-axes. We order the quadrilaterals $q_1, q_2, \ldots, q_h$ that contain w in a counterclockwise manner, as shown in Figure 4. Let w, x, y, z be the vertices in counterclockwise order of a quadrilateral q containing w; the interior of q lies to the left as the edges of q are traversed in order. There are three types of quadrilaterals. (See Figure 4.)

Type 0: Neither x nor z lies on segment wy.

Type 1: Point x lies on segment wy.

Type 2: Point z lies on segment wy.

Observation 1: If the point p in P_n is in the angle determined by the rays yx and yz, then every point in quadrilateral q is visible from p.

Now Observation 1 implies that if p is any point in Quadrant I that is sufficiently close to w, then every point in a quadrilateral of type 0 is visible from p. The degenerate quadrilaterals of types 1 and 2 place further restrictions on our desired set R_w, which are captured by the following observation.

Observation 2: There exists a nonempty region R_w with the desired visibility property provided every quadrilateral of type 1 occurs before the first quadrilateral of type 2 in the list $q_1, q_2, \ldots, q_h$.

We now show that no quadrilateral of type 2 precedes a quadrilateral of type 1, which will complete the proof of the lemma and of Theorem 1. Partition the vertices of P_n into the alternating sets V^+ and V^-, as in the proof of Proposition 1. Without loss of generality $w \in V^+$.

Observation 3: In a counterclockwise traversal of the boundary of the polygon P_n each vertex in V^+ is entered horizontally and exited vertically, while each vertex in V^- is entered vertically and exited horizontally.

Claim 1: The line segment wy cannot have negative slope in a quadrilateral q of type 1 or 2. For suppose that vertex y is in Quadrant IV and q is of type 1, as shown in Figure 5(a). Then $x \in V^-$, and hence x is entered vertically and is exited horizontally along the boundary of P_n. But then the interior angle at x must be greater than 270°, which is impossible. The argument is similar when q is of type 2 and when y is in Quadrant II.

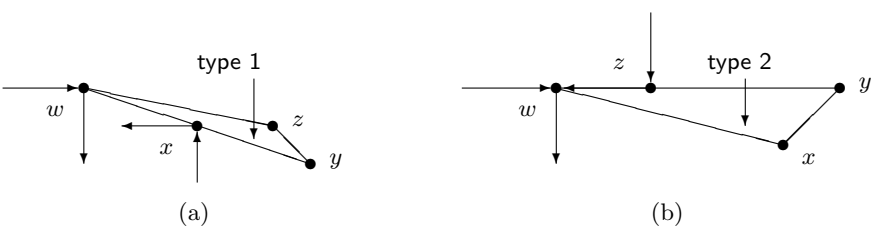


Fig. 5. (a) The proof of Claim 1 (b) The proof of Claim 2

Claim 2: Vertices z and y cannot be on the positive x-axis in a quadrilateral of type 2. For suppose we have such a quadrilateral, as in Figure 5(b). Then $z \in V^-$, and it follows that z is entered from above and is exited to the left. Let z' be the point in V^- along segment wz that is closest to w. Then $z'w$ must be a boundary edge of Q_n, and so w meets three boundary edges, which is impossible.

In a similar manner one shows that x and y cannot be on the positive y-axis in a quadrilateral of type 1.

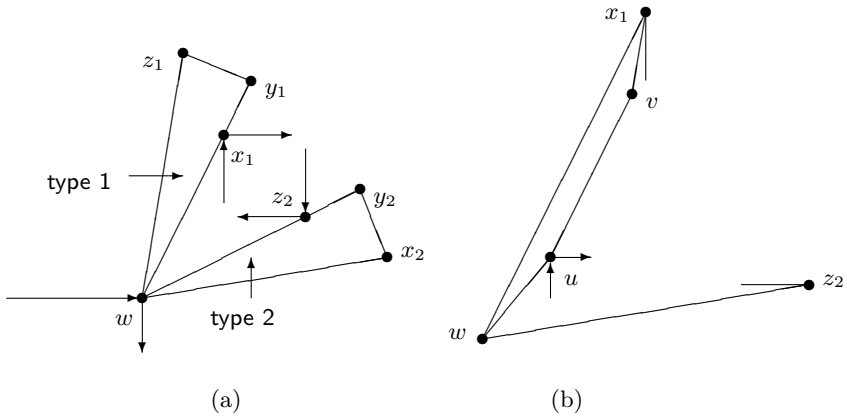


Fig. 6. (a) A quadrilateral of type 2 cannot precede a quadrilateral of type 1 (b) The proof of Lemma 2

Now assume that a quadrilateral of type 2 with vertices w, x_2, y_2, z_2 precedes a quadrilateral of type 1 with vertices w, x_1, y_1, z_1 in the list $q_1, q_2, \ldots, q_h$. Then our claims imply that points y_1 and y_2 are both in the interior of Quadrant I and that segment wy_1 is above segment wy_2, as in Figure 6(a). Also, Observation 3 implies that in a counterclockwise traversal of P_n vertex x_1 must be entered from below and exited to the right, and vertex z_2 must be entered from above and exited to the left. Now the diagonals wx_1 and wz_2 partition P_n into three polygons, each of which has a convex quadrangulation. Let P_m denote the polygon that has x_1, w, and z_2 as consecutive vertices. Then the angles at x_1, w, and z_2 in P_m must be acute. Thus P_m has a convex quadrangulation and each interior angle is either 90° or 270°, except for the three consecutive acute angles at x_1, w, and z_2. The following lemma proves that such a polygon does not exist. □

Lemma 2. *Let P_m be a polygon with each interior angle equal to* 90° *or* 270°, *except for three consecutive acute angles. Then P_m does not have a convex quadrangulation.*

Proof. Assume that P_m does have a convex quadrangulation. We obtain a contradiction by induction. Note that m must be even. Suppose that $m = 4$. Then the one non-acute angle of P_m must equal 270°, rather than 90°, for the sum of

the four angles to equal 360°. A quadrilateral with a 270° angle does not have a convex quadrangulation.

Now suppose that $m \geq 6$. We continue the notation from Lemma 1 and let the three acute angles be at vertices x_1, w, and z_2, as in Figure 6(b). We claim that the sum a of these three acute angles must be 90°. For let P_m contain r angles equal to 270°. Then $180(m-2) = 270r + (m-3-r)90 + a$, and thus $a = 90(m-2r-1)$. We know that m is even and that $a < 270$. The only possibility is $a = 90$.

We partition the vertices of P_m into two alternating sets V^+ and V^-, as before, with $w \in V^+$, and we orient the edges of P_m counterclockwise so that the interior of P_m lies to the left of each edge. Each vertex in V^- is exited horizontally (except for x_1) and is entered vertically (except for z_2). Now let the convex quadrilateral q containing side x_1w of P_m have vertices w, u, v, x_1 in counterclockwise order. The sum of the angles in q is 360°, and the angles in q at w and x_1 sum to less than 90°. Neither of the angles in q at u and v can be greater than 180°. It follows that the angles in q at u and v must be greater than 90°, and therefore the angles at u and v in the polygon P_m must equal 270°.

Now $u \in V^-$ and $u \notin \{x_1, z_2\}$. Therefore u is entered vertically and is exited horizontally in a counterclockwise traversal of the boundary of P_m. The only possibility is that u is entered from below and is exited to the right. Now the diagonal wu partitions P_m into two smaller polygons each of which has a convex quadrangulation. One of these smaller polygons contains three consecutive acute angles at u, w, and z_2, with all other angles equal to 90° or 270°. This contradicts the inductive hypothesis. □

References

1. V. Chvátal, A combinatorial theorem in plane geometry, *J. Combin. Theory Ser. B*, **18** (1975), 39–41.
2. S. Fisk, A short proof of Chvátal's watchman theorem, *J. Combin. Theory Ser. B*, **24** (1978), 374.
3. J. Kahn, M. Klawe, and D. Kleitman, Traditional galleries require fewer watchmen, *SIAM J. Alg. Disc. Meth.*, **4** (1983), 194-206.
4. B.-C. Liaw, N.F. Huang, and R.C.T. Lee, The minimum cooperative guards problem on k-spiral polygons (Extended Abstract), in *Proc. 5-th Canadian Conf. on Computational Geometry (5CCCG)*, Waterloo, Ontario, Canada, (1993), 97–102.
5. B.-C. Liaw and R.C.T. Lee, An optimal algorithm to solve the minimum weakly cooperative guards problem for 1-spiral polygons, *Inform. Process. Lett.*, **57** (1994), 69–75.
6. T.S. Michael and V. Pinciu, Art gallery theorems for weakly cooperative guards, submitted.
7. J. O'Rourke, *Art Gallery Theorems.* Oxford University Press, 1987.
8. T.C. Shermer, Recent results in art gallery theorems, *Proc. IEEE,* **80** (1992), 1384–1399.

Reachability on a region bounded by two attached squares

Ali Mohades
mohades@cic.aku.ac.ir
AmirKabir University of Tech., Math. and Computer Sc. Dept.

Mohammadreza Razzazi
razzazi@ce.aku.ac.ir
AmirKabir University of Tech., Computer Eng. Dept.

Abstract

This paper considers a region bounded by two attached squares and a linkage confined within it. By introducing a new movement called *mot*, presents a quadratic time algorithm for reaching a point inside the region by the end of the linkage. It is shown that the algorithm works when a certain condition is satisfied.

keywords: Multi-link arm, reachability, motion planning, concave region, robot arms.

1 Introduction

This paper considers the movement of a linkage in a two-dimensional bounded region and introduces a new algorithm to reach a given point by the end of the linkage. The region considered is the one obtained by two attached squares.

Several papers have been written on reachability problems mainly, on convex region. Hopcroft, Joseph and Whitesides in [1] studied the reconfiguration and reachability problems for a linkage. In [2], they gave a polynomial time algorithm for moving a linkage confined within a circle from one given configuration to another, and proved that the reachability problem for a planar arm constrained by an arbitrary polygon, is NP-hard. Joseph and Plantings [3] proved that the reachability problem for a chain moving within a certain non-convex constraining environment is PSPACE hard.

In [4] and [5], Kantabutra presented a linear time algorithm for reconfiguring certain chains inside squares. He considered an unanchored n-linkage robot arm confined inside a square with side length at least as long as the longest arm link and found a necessary and sufficient condition for reachability in this square. His algorithm requires O(n) time.

This paper extends the previous results by providing a quadratic time algorithm to solve the reachability problem in a special concave region. The

V.N. Alexandrov et al. (Eds.): ICCS 2001, LNCS 2073, pp. 763–771, 2001.
©Springer-Verlag Berlin Heidelberg 2001

region is bounded by the union of two squares attached via one edge. In the next section of the paper some preliminaries and useful definitions are given. In section 3 a new movement, by which a linkage moves in a concave corner is formulated and finally in section 4 present the reachability algorithm and the related properties are presented.

2 Preliminaries

An *n-linkage* Γ[0,1,...n] is a collection of n rigid rods or links, $\{A_{i-1}A_i\}_{i=1,...n}$, consecutively joined together at their end points, about which they may rotate freely. Links may cross over one another and none of end points of the linkage are fixed.

We denote the length of links of Γ[0,1,...n] by $l_1, l_2, ...l_n$, where l_i is the length of link with end points A_{i-1} and A_i and $||\Gamma|| = max_{1\leq i\leq n} l_i$. For $1 \leq i \leq n-1$ the angle obtained by turning clockwise about A_i from A_{i-1} to A_{i+1} is denoted by α_i. We say that a linkage Γ is *bounded by b* if $||\Gamma|| <$ b, i.e no link has a length greater than or equal to b.

For a region P, by *Reaching a given point p* $\in P$ *by* A_n, the end point of Γ, we mean Γ can move within P from its given initial position to a final position so that A_n reaches p.

For a linkage Γ confined inside a convex region P with boundary denoted by ∂P, we define two special configurations as follows (Figure 1):

We say that Γ is in *Rim Normal Form* (denoted RNF), if all its joints lie on ∂P.

We say that Γ is in *Ordered Normal Form* (denoted ONF), if:

1. Γ is in RNF.
2. Moving from A_0 toward A_n along Γ is always either clockwise or counterclockwise around the boundary polygon.

Algorithms for the reconfiguration of an n-linkage usually break up the motions for the whole reconfiguration into simple motions, in which only a few joints are moved simultaneously (see [2], [6] and [7]). We allow the following type of simple motions:

- No angle at joints changes, but the linkage may translate and rotate as a rigid object.
- At most four angles change simultaneously and the other joints do not change their positions.

3 Movement in a concave environment

In this section we introduce a new movement for a linkage to reach a point inside a certain concave region.

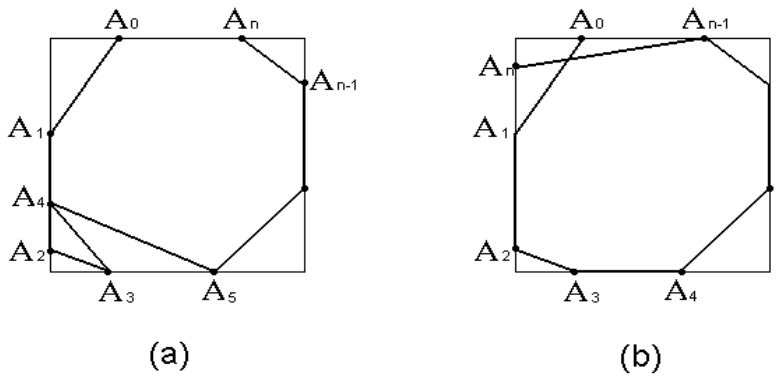


Figure 1: *An n-linkage in (a): Rim Normal Form, (b): Ordered Normal Form.*

Theorem 1. Suppose that S is a region where its boundary polygon ∂S, is a square with side length s, $\Gamma[0, 1, ...n]$ is an n-linkage confined within S and $\|\Gamma\| <$ s. Then Γ can be brought to ONF using O(n) simple motions.

Proof: See [5].

Lemma 2. If ∂S, the boundary polygon of the region S, is a square with side length s and $\Gamma[0, 1, ...n]$ is an n-linkage with $\|\Gamma\| <$ s confined within S, initially in ONF. Then any joint of Γ can be moved along ∂S in either direction, in such a manner that the linkage always remain in ONF. This can be done with O(n) simple motions.

Proof: See [5].

To understand our new movement, it helps to first consider a special case of 2-linkage $\Gamma[1, 2, 3]$ consisting of joints A_1, A_2 and A_3. We define a movement for $\Gamma[1, 2, 3]$ from its initial configuration to a specified final configuration in which, A_1 gets the position of A_2, and A_3 moves forward in a given path (Figure 2).

Unless otherwise specified, by $\angle A_1A_2A_3$ ($\angle\gamma_1\gamma_2$, which γ_1 and γ_2 are two crossing line segments), we mean the angle obtained by turning clockwise from A_1 to A_3 about A_2 (from γ_1 to γ_2).

Circumstances: Consider two line segments γ_1 and γ_2 which intersect at q and $\angle\gamma_1\gamma_2$ is in $[\pi, 2\pi]$. Let ρ be the line segment which starts at q and divides the angle $\angle\gamma_1\gamma_2$ into two angles $\angle\gamma_1\rho$ and $\angle\rho\gamma_2$ in such a way that $\angle\gamma_1\rho$ is in $[\pi/2, \pi]$. Initial configuration of Γ[1,2,3] is defined as follows: Let A_1 be at point p on line segment γ_1, A_2 at q and A_3 at point r on line segment γ_2 (Figure 2-a). By this assumption we can define our movement in a concave region.

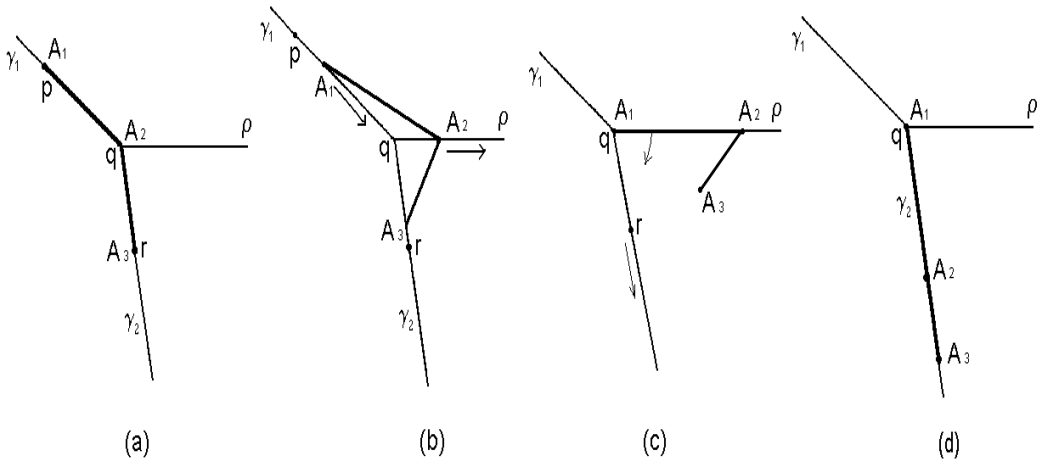


Figure 2: *(a): Initial configuration of* $\Gamma[1,2,3]$, *(b): middle-joint-up*(A_1, A_2, A_3, ρ) *motion, (c): front-link-forward*(A_1, A_2, A_3, ρ) *motion, (d): final configuration of* $\Gamma[1,2,3]$.

Definition 3. The *mot*(A_1, A_2, A_3, ρ) movement changes the initial configuration of $\Gamma[1,2,3]$ to a final configuration by which Γ lies on γ_2. This is done by two consecutive motions:

- *Middle-joint-up*(A_1, A_2, A_3, ρ): moves A_2 along ρ away from q until A_1 reaches q. During the movement A_1 remains on γ_1, and A_3 remains on γ_2 as much as possible.
- *Front-link-forward*(A_1, A_2, A_3, ρ): fixes A_1 at q and brings down A_3 on γ_2 (if not already there). To straighten Γ, it moves A_3 along γ_2 away from q.

We show the mot(A_1, A_2, A_3, ρ) movement can be done in finite number of simple motions.

Assume Γ is in the initial configuration. We show how each of the middle-joint-up motion and front-link-forward motion is done in finite number of simple motions.

Middle-joint-up(A_1, A_2, A_3, ρ):
Move A_2 along ρ away from q (Figure 2-b). If $\angle\rho\gamma_2 \geq \pi/2$, during the movement, A_1 and A_3 approach q, while staying on lines γ_1 and γ_2 respectively.

If $\angle\rho\gamma_2 < \pi/2$, during the movement, A_3 moves away from q and it is possible that A_2A_3 becomes perpendicular to γ_2. If this happens, first turn A_2A_3 about A_2 until qA_2A_3 folds, then if needed, move A_2A_3 along ρ away from q in a way that α_2 increases until $A_1A_2A_3$ folds and A_1 reaches q. This requires a finite number of simple motions.

Front-link-forward(A_1, A_2, A_3, ρ):
If during middle-joint-up motion A_1 reaches q first, for applying front-link-forward motion, it is enough to keep A_1 at q fixed, and move A_3 along γ_2 until Γ straightens.

If A_3 reaches q first and A_1 arrives later, for applying front-link-forward motion, turn A_2A_3 about A_2 in a way that α_2 decreases, until A_3 hits γ_2 or $\alpha_2 = 3\pi/2$. If $\alpha_2 = 3\pi/2$ before A_3 hits γ_2, rotate Γ about A_1 in a way that $\angle A_2A_1r$ decreases until A_3 reaches γ_2, then keep A_1 fixed at q and move A_3 along γ_2 away from q so that Γ straightens. This requires a finite number of simple motions (Figure 2-c).

If A_3 hits γ_2 first, keep A_1 fixed at q and move A_3 along γ_2 away from q so that Γ straightens.

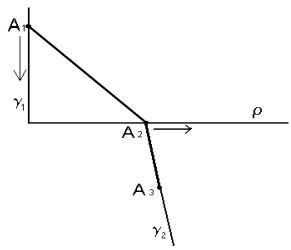


Figure 3: γ_1 *can be a convex path instead of a line segment.*

In the definition 3, during mot(A_1, A_2, A_3, ρ) movement, A_1 moves along the line segment γ_1. The line segment γ_1 can be replaced by a composition of two line segments in such a way that the path where A_1 belongs to is convex. See figure 3.

In our algorithm, to reach p we have to apply mot$(A_{i-1}, A_i, A_{i+1}, \rho)$ movement several times. At the end, possibly p can be reached by A_n somewhere during one of the middle-joint-up or the front-link-forward. It means that algorithm stops before the last mot$(A_{i-1}, A_i, A_{i+1}, \rho)$ movement is terminated. Such a movement is called *partial-mot*$(A_{i-1}, A_i, A_{i+1}, \rho)$ movement. This is a movement in according with the mot$(A_{i-1}, A_i, A_{i+1}, \rho)$ movement, the movement stops somewhere during one of the middle-joint-up or the front-link-forward motion in such a way that A_3 remains on γ_2.

4 The reachability algorithm

In this section, we study reachability in a region bounded by two squares in which the whole or a part of a side of one square coincides with a part of a side of the other.

Assume S_1 and S_2 are two regions bounded by squares ∂S_1 and ∂S_2 with side lengths s_1 and s_2 respectively. Let squares ∂S_1 and ∂S_2 be attached via one side (the whole or a part of a side) and S $= S_1 \cup S_2$. Let $\Gamma = [0, 1, ...n]$ be an n-linkage confined within S_1 (Figure 4-a). In the following theorem we explain how A_n, the end of Γ, can reach a point p $\in S_2$.

Let ρ be the line segment shared by S_1 and S_2 and let v_1 and v_2 be two end points of ρ, where v_1 is the farthest point of ρ from p (Figure 4-b).

The following theorem presents sufficient condition for reachability of a given point in S by the end of a linkage confined within S.

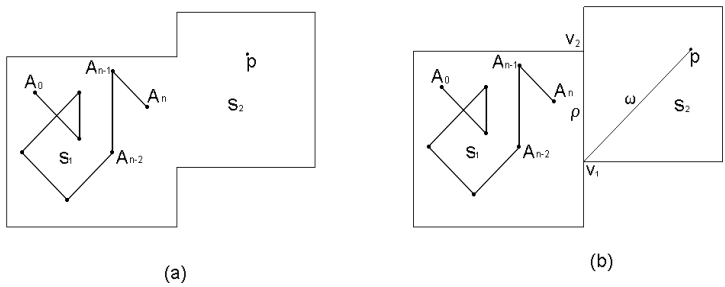


Figure 4: Γ *confined within* S_1 *and* $p \in S_2$.

Theorem 4. Suppose $p \in S_2$, Γ confined within S_1, $\|\Gamma\| < \text{Min}\{\frac{\sqrt{2}}{2}s_1, \|\rho\|\}$, then with O($n^2$) simple motions − in the worst case − p can be reached by A_n.

Proof: We introduce an algorithm to bring A_n to p using O(n^2) simple motions, in the worst case.

Assume that ω is the line including v_1p, and moving from v_2 to v_1 on the side of ∂S_1 which includes v_2 and v_1 is clockwise. At the beginning we bring Γ to ONF in S_1. By theorem 1, this is done in O(n) simple motions. Without loss of generality we assume that Γ is placed on ∂S in counterclockwise order of indices of links'joints. Then Γ is moved along ∂S_1 counterclockwise until A_n reaches v_1. This can be done while no joint of Γ leaves ∂S_1.

We consider two cases: $d(p, v_1) \geq ||A_{n-1}A_n||$ and $d(p, v_1) < ||A_{n-1}A_n||$.

Case 1: $d(p, v_1) \geq ||A_{n-1}A_n||$. The algorithm consists of three steps. In the first step A_n is brought into S_2. In the second step Γ is moved so that $\Gamma[0, k_0]$ takes ONF in S_1 (k_0 will be defined in step 2), A_{k_0} coincides with v_1, and $\Gamma[k_0, n] \subset \omega$, and finally, in the last step A_n reaches p.

Step 1: Move Γ along ∂S_1 counterclockwise until A_{n-1} reaches v_1, because $\|\Gamma\| < \|\rho\|$, A_n doesn't pass v_2, this takes O(n) (Figure 5-a). Then rotate A_n clockwise about $A_{n-1} = v_1$ toward ω until A_n lies on ω. If $d(p, v_1) = ||A_{n-1}A_n||$, A_n reaches p and we are done. If not, we pass to the second step. This step takes O(n).

Step 2: We define $k_0 = \min\{\text{k } | d(p, v_1) \geq \sum_{i=k+1}^{n} l_i\}$. Since $d(p, v_1) \geq l_n$, then $k_0 \leq n - 1$. Suppose that, for $j > k_0$, $\Gamma[j, n] \subset \omega$ is straight, A_j coincides with v_1, and $\Gamma[1, j]$ gets ONF in S_1, by using mot$(A_{j-1}, A_j, A_{j+1}, \rho)$,

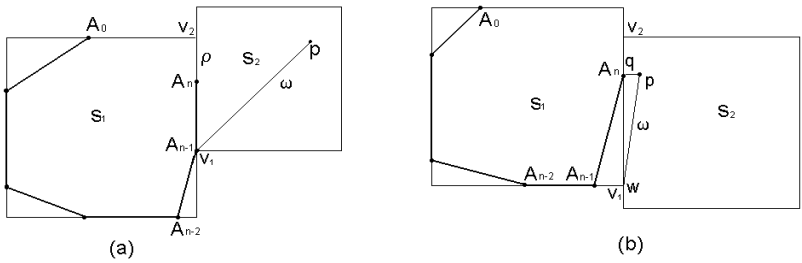


Figure 5: *(a):* $d(p, v_1) > \|A_{n-1}A_n\|$*, (b):* $d(p, v_1) < \|A_{n-1}A_n\|$ *and* $v_1 = w$

Γ is moved to a configuration in which $\Gamma[j-1, n] \subset \omega$ straightens, A_{j-1} coincides with v_1, and $\Gamma[1, j-1]$ is in ONF in S_1.

By repeating this process, Γ can move to a configuration in which, $\Gamma[1, k_0]$ gets ONF, A_{k_0} coincides with v_1, and $\Gamma[k_0, n] \subset \omega$.

If $k_0 > 0$, since $\sum_{i=k_0}^{n} l_i > d(p, v_1) > \sum_{i=k_0+1}^{n} l_i$, A_n reaches p during $\mathrm{mot}(A_{k_0-1}, A_{k_0}, A_{k_0+1}, \rho)$. Therefore we move Γ according to $\mathrm{partial\text{-}mot}(A_{k_0-1}A_{k_0}, A_{k_0+1}, \rho)$, depending on values of $\angle v_2 v_1 p$, l_{k_0} and $d(p, v_1)$, A_n reaches p during one of the middle-joint-up motion or the front-link-forward motion. This step takes $\mathrm{O}(k_0 \mathrm{n})$ and is $\mathrm{O}(n^2)$ in the worst case.

If $k_0 = 0$, A_n doesn't reach p during this step and we pass to step 3.

Step 3: In the case of $k_0 = 0$, i.e. $\sum_{i=1}^{n} l_i < d(p, v_1)$, by step2, Γ may move to a configuration in which, A_0 coincides with v_1 and $\Gamma \subset \omega$ straightens. It is enough to move Γ along ω toward p until A_n reaches p. This step takes O(1).

Case 2: $d(p, v_1) < \|A_{n-1}A_n\|$. Assume that ω intersects ∂S_1 at w (it is possible that w may coincides with v_1 (Figure 5-b)). Let the circle $C(v_1, \|pv_1\|)$ intersect v_1v_2 at q. To reach p, move Γ counterclockwise along ∂S_1 until A_n reaches q. Depending on the position of A_{n-1} on ∂S_1 one of the three following subcases occurs.

Subcase 2.1: A_{n-1} resides on the side of ∂S_1 containing v_1v_2. In this situation v_1 belongs to the link $A_{n-1}A_n$ and $C(p, l_n)$ intersects the line segment ω at point g . Rotate $A_{n-1}A_n$ clockwise about v_1 toward p. Because $\|\Gamma\| < \frac{\sqrt{2}}{2}s_1$, $C(g, l_{n-1})$ cannot contain S_1 i.e. A_{n-2} does not need to exit S_1. Continue rotation until A_{n-1} reaches g and A_n reaches p. During rotation, A_{n-1} exits ∂S_1 and if $C(g, l_{n-1})$ intersects ∂S_1, A_{n-2} can be stayed on ∂S_1 and $\Gamma[0...n-2]$ remains in ONF (Figure 6-a).

Otherwise if $C(g, l_{n-1})$ does not intersect ∂S_1, consider the largest $0 < k_0$ in such a way $C(g, l_{n-1}... + l_{k_0})$ intersects ∂S_1, otherwise let $k_0 = 1$. During rotation we let A_{n-1},...,A_{k_0} exit ∂S_1 while making $\alpha_{n-1} = ... = \alpha_{k_0+1} = \pi$, keeping $\Gamma[k_0...n-1]$ straight and remaining $\Gamma[0...k_0]$ in ONF.

Subcase 2.2: A_{n-1} resides on the side of ∂S_1 adjacent to the side containing v_1v_2, and ω intersects link A_nA_{n-1}. To reach p, first fix $\Gamma[0, 1, ...n-1]$ and rotate $A_{n-1}A_n$ about A_{n-1} toward p until link $A_{n-1}A_n$ reaches v_1. Then rotate $A_{n-1}A_n$ about v_1 toward ω until A_n hits ω. During rotation A_n does not hit ∂S_1. Finally slip $A_{n-1}A_n$ on ω until A_n reaches p. During the move-

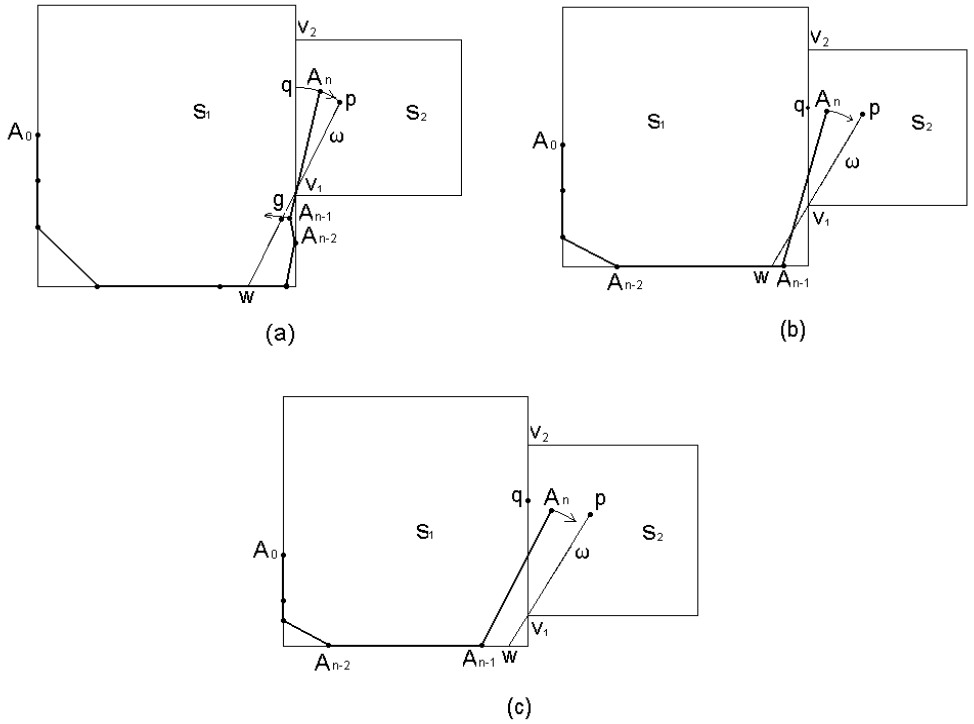


Figure 6: *(a): A_{n-1} belongs to the same edge as v_1, (b): A_n and A_{n-1} are in both sides of ω, (c): A_n and A_{n-1} are in the same side of ω*

ment, one of the possibilities similar to the previous situation will happen, which can be treated accordingly (Figure 6-b).

Subcase 2.3: Like case 2.2, but ω does not intersect link A_nA_{n-1}. Suppose that $C(p, l_n)$ intersects ∂S_1at g. i.e. p is visible from g. To reach p, first fix $\Gamma[0, 1, ...n-1]$ and rotate $A_{n-1}A_n$ about A_{n-1} toward ω until A_n reaches ω. Then, move A_n along ω toward p. During movement $\Gamma[0, 1, ...n-1]$ does not exit ∂S_1 and A_n gets to p while A_{n-1} reaches g. Refer to Figure 6-c.

Each of these subcases takes O(n).

References

[1] J. Hopcroft, D. Joseph and S. Whitesides. *Movement problems for 2-dimensional linkages.* SIAM J. Compt., 13: pp. 610-629, 1984.

[2] J. Hopcroft, D. Joseph and S. Whitesides. *On the movement of robot arms in 2-dimensional bounded regions.* SIAM J. Compt., 14: pp. 315-333, 1985.

[3] D. Joseph and W.H. Plantings. *On the complexity of reachability and motion planing questions.* Proc. of the symposium on computational geometry. ACM, June 1985.

[4] V. Kantabutra. *Motions of a short-linked robot arm in a square.* Discrete and Compt. Geom., 7:pp. 69-76, 1992.

[5] V. Kantabutra. *Reaching a point with an unanchored robot arm in a square.* International jou. of comp. geo. & app., 7(6):pp. 539-549, 1997.

[6] W.J. Lenhart and S.H. Whitesides. *Reconfiguration using line tracking motions.* Proc. 4th Canadian Conf. on computational geometry, pp. 198-203, 1992.

[7] M. van Krevel, J. Snoeyink and S. Whitesides. *Folding rulers inside triangles.* Discrete Compt. Geom., 15:pp. 265-285, 1996.

Illuminating Polygons with Vertex π-Floodlights

Csaba D. Tóth*

Institut für Theoretische Informatik
ETH Zürich, CH-8092 Zürich
Switzerland
toth@inf.ethz.ch

Abstract. It is shown that any simple polygon with n vertices can be illuminated by at most $\lfloor (3n-5)/4 \rfloor$ vertex π-floodlights. This improves the earlier bound $n-2$, whereas the best lower bound remains $3n/5+c$.

1 Introduction

The first theorem on *Art Galleries* is due to Chvatál [1] who showed that any simple polygon with n vertices can be illuminated by $\lfloor n/3 \rfloor$ light sources and this bound is tight. The famous proof of Fisk [4] places light sources at vertices of the polygon. It has been shown recently [7], that $\lfloor n/3 \rfloor$ is sufficient even if the light sources can illuminate only a range of angle π (i.e. using π-floodlights). But there, π-floodlights may be placed at any point of the polygon, even two π-floodlights are allowed to be placed at the same point.

Urrutia [2] asked the following question. What is the minimal number of vertex π-floodlights that can collectively illuminate any simple polygonal domain (shortly *polygon*) P with n vertices. A vertex π-floodlight is given by a pair (v, H^v) where v is a vertex of P and H^v is a closed half-plane H^v such that v is on the boundary of H^v. There may be at most one π-floodlight at each vertex of P. A π-floodlight at (v, H^v) *illuminates* $a \in P$ if and only if the closed line segment va is in $P \cap H^v$. All points of P should be *illuminated* by at least one π-floodlight.

F. Santos [9] has produced a family of polygons that requires $\lfloor 3n/5 \rfloor + O(1)$ vertex π-floodlights. Urrutia [2] conjectured that this number is always sufficient to illuminate any polygon with n vertices but proved only the sufficiency of $n-2$. So far no constant $b < 1$ has been known such that $bn+O(1)$ vertex π-floodlights can illuminate any polygon with n vertices.

Theorem 1. $\lfloor 3(n-3)/4 \rfloor + 1$ *vertex π-floodlights can illuminate any simple polygon with n vertices.*

The notion of vertex α-floodlight can be defined for any angle $0 < \alpha < 2\pi$ as a cone of aperture at most α with apex at a vertex of polygon P. Under the

* The author acknowledges support from the Berlin-Zürich European Graduate Program "Combinatorics, Geometry, and Computation".

V.N. Alexandrov et al. (Eds.): ICCS 2001, LNCS 2073, pp. 772–781, 2001.
© Springer-Verlag Berlin Heidelberg 2001

condition that there may be at most one vertex floodlight at each vertex, it is known [2] that for any angle $\alpha < \pi$, there exist convex polygons P_n with $n \geq n_\alpha$ vertices such that n α-floodlights cannot illuminate P_n.

In this paper, the placement of floodlights is based on a decomposition of the polygon into "dense polygons". Such decomposition was introduced in [8], and is discussed in our Sect. 2. Any dense polygon with n vertices can be illuminated with at most $\lfloor (3n-5)/4 \rfloor$ vertex π-floodlights. This does not imply immediately that any polygon P could be illuminated by $3n/4 + O(1)$ floodlights, because at most one vertex π-floodlight can be placed at each vertex of P, and thus there may be conflicts at vertices belonging to several dense sub-polygons. Our floodlight placement algorithm and its analysis is contained in Sect. 4.

2 Dense Polygons

Let P be a simple polygon and let T be a set of triangles in the plane. T is the triangulation of P, if $P = \bigcup T$, the triangles of T are pairwise non-overlapping, and the vertices of the triangles are vertices of P. It is known that every simple polygon has a triangulation, every triangulation consists of exactly $n-2$ triangles, although the triangulation is not necessarily unique. We define the graph $G(T)$ on a triangulation T. The nodes of the graph correspond to the elements of T, two nodes are adjacent if and only if the corresponding triangles have a common side. $G(T)$ is a tree on $n-2$ nodes, the maximal degree in $G(T)$ is three, since a triangle $t \in T$ may have a common side with at most three other triangles of T.

Definition 1. *A graph G is dense, if G is a tree and each node of G has degree one or three. A simple polygon S is dense if graph $G(T_S)$ is dense for any triangulation T_S of S.*

Proposition 1. *Any dense graph G has an even number of nodes. If a dense graph G has 2ℓ nodes then it has exactly $\ell + 1$ leaves.*

Proof. Suppose that G has k leaves and l nodes of degree 3. The number of edges is $(k+3l)/2 = (k+l)/2+l$, hence $k+l$ is even. G is a tree, so $(k+l)/2+l = k+l-1$, that is $k = (k+l)/2+1$. □

2.1 Dense Decomposition

The *dense decomposition* L of a polygon P is set of pairwise non-overlapping dense polygons such that $P = \bigcup L$ and the vertices of the dense polygons are vertices of P. We can define the tree $G(L)$ on a dense decomposition L just like $G(T)$. The nodes of the G(L) correspond to the dense polygons of L, two nodes are adjacent if and only if the corresponding polygons have a common side.

The union of the triangulations of elements of a dense decomposition is a triangulation T of P. So T contains an even number of triangles. Clearly, this is impossible if P has an odd number of vertices. We can prove, however, the following Lemma.

Lemma 1. *If P is a simple polygon with an even number of vertices, then P has a dense decomposition.*

Proof. By induction on the number of nodes of $G(T)$. Every quadrilateral is dense. If the polygon P is dense, then the proof is complete.

If P is not dense then there is a triangulation T of P such that there exists a node of degree two in $G(T)$. Consider the tree $G(T)$ as a rooted tree $(G(T), r)$ where an arbitrary leaf r of $G(T)$ is chosen as root. Let $v \in G(T)$ be a node of degree two such that no descendant of v has degree two in $G(T)$. Let S denote the subtree containing v and all its descendant in $(G(T), r)$. According to Proposition 1, the subtree S has an even number of nodes, hence $G(T) \setminus S$ has an even number of nodes as well. The polygons corresponding to S and $G(T) - S$ have dense decompositions by induction, and together they give a dense decomposition of P. □

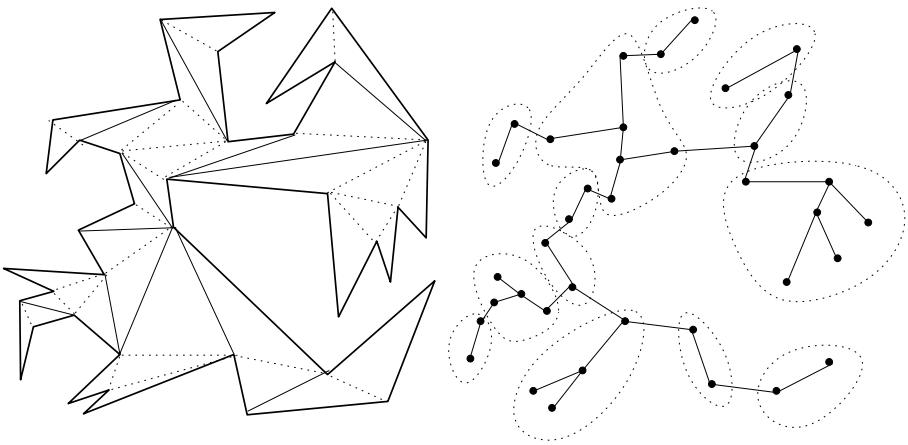


Fig. 1. Dense decomposition of a polygon P and the corresponding graph G

If P has an odd number of nodes, then let t correspond to a leaf of $G(T)$ in a triangulation T of P. Polygon $P - t$ has already an even number of vertices, and t can be illuminated by one π-floodlight at the unique vertex not adjacent to $P - t$. To establishes Theorem 1, it is enough to prove the following.

Lemma 2. *Any simple polygon P with an even number n of vertices can be illuminated by $\lfloor 3(n-2)/4 \rfloor$ vertex π-floodlights.*

2.2 Notation of Dense Polygons

Let L be the dense decomposition of a simple polygon. Consider G(L) as a rooted tree $(G(L), r)$ with an arbitrary leaf r chosen as root. In this way we may

interpret the parent-son relation between dense polygons of L. For any polygon $S \in L$, $S \neq r$, let the *base side* of L be the side adjacent to the parent polygon of S. For the root polygon, let the *base side* be any side not adjacent to any son polygon. A base vertex of a polygon $Q \in L$ is a vertex along its base side.

In our method, *convex* and *concave* quadrilaterals of the dense decomposition have different roles. We call *star-polygon* a dense polygon on at least 6 vertices. Fix a triangulation T_S in each $S \in L$.

An *outer vertex* of a star-polygon or concave quadrilateral $S \subset P$ is a vertex of S which belongs to exactly one triangle $t \in T_S$. All other vertices of a star polygon are called *inner*. Every vertex of a convex quadrilateral in L is *outer*. The *corner* vertex of a concave quadrilateral $Q \in L$ is the vertex opposite to the reflex vertex of Q (a π-floodlight at v can illuminate Q).

Proposition 2.
(1) For every vertex v of P, there is at most one dense polygon $S \in L$ such that v is a non-base vertex of S.
(2) Every second vertex of a dense polygon is outer.
(3) Every outer vertex of a dense polygon is convex.
(4) A star polygon or a concave quadrilateral of L with 2ℓ triangles has exactly $\ell + 1$ outer vertices, one of which is a base vertex.

The proof of the above statements is immediate.

Proposition 3. *For two vertices v and x of a star-polygon S, vx cannot be a diagonal of S if v is an outer vertex for a triangulation T_S.*

Proof. Let u, v, and w be three consecutive vertices of S such that uvw is a triangle in a triangulation T_S of S. Suppose that vx is a diagonal of S. First we state that there is a vertex y such that both uvy and uvy are diagonals. If $y = x$ does not have this property, then let y be the vertex in uvx where the angle $\angle uvy$ is minimal. There is a triangulation T'_S of S such that $uvy \in T'_S$. In $G(T')$, the node corresponding to uvy has degree 2. □

3 Illuminating Star Polygons

In our method, every dense polygon $S \in L$ will be illuminated by π-floodlights placed at vertices of S. Lemma 2 is best possible in this setting, because there are dense hexagons that cannot be illuminated by less than three vertex π-floodlights.

Definition 2. *A π-floodlight (v, H^v) in P is called complementary, if P has a reflex angle α at v and the angular domain of α contains the complementer of the closed half-plane H^v.*

Lemma 3. *Any dense polygon $S \in L$ with 2ℓ vertices can be illuminated by at most ℓ vertex π-floodlights at vertices of S: one at an arbitrary outer vertex and at most $\ell - 1$ complementary π-floodlights at reflex vertices.*

Proof. Fix a triangulation T of S. Suppose that we put a floodlight at an outer vertex v. v belongs to a unique triangle $xvy \in T$. Let $C(v)$ be the set of points $p \in S$ such that the line segment vp is in S. The rays $\overrightarrow{vx}$ and $\overrightarrow{vy}$ hit the boundary of S at points x' and y' resp.

According to Proposition 3, x' and y' are points of a same side ab of S. We may suppose that b is an outer vertex. $S \setminus C(v)$ consists of at most two simple polygons S_x and S_y such that $x \in S_x$ and $y \in S_y$. Every reflex vertex of S is a vertex of exactly one of S_x and S_y. We may suppose w.l.o.g. that S_x is non-empty.

Visit the reflex angles of S_x along the boundary from x to a in orientation yvx. Consecutively dissect S at each reflex vertex w by a ray emanating from w such that angle $\angle w > 180°$ is partitioned into $180°$ and $\angle w - 180°$ (e.g. $\angle x$ is dissected by the segment xx'). Repeat this dissection in S_y as well if $S_y \neq \emptyset$. Thus $S \setminus C(v)$ is partitioned into k convex polygons where k is the number of reflex angles in S.

Our proof is complete if S has at most $\ell - 1$ reflex angles (e.g., S is quadrilateral). Suppose that S has ℓ reflex angles, hence x is also reflex vertex. We state that the last convex region C (along ax') is already illuminated by a complementary floodlight at another reflex vertices of S_x. (See Fig. 2 for illustrations.)

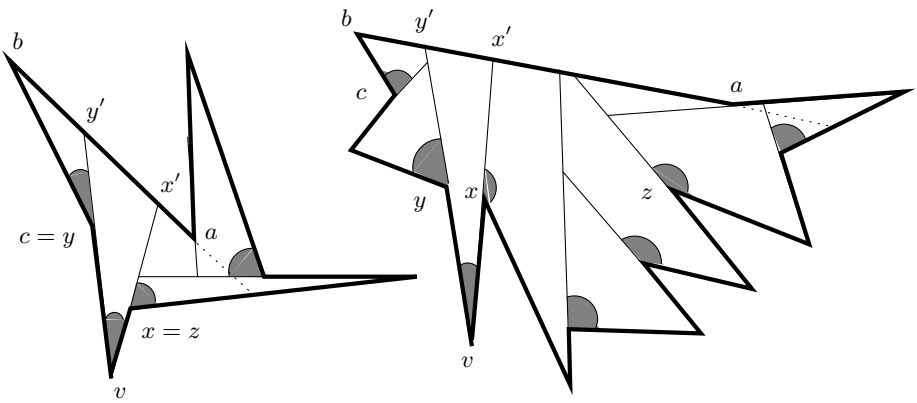


Fig. 2. Illuminating star polygons with 8 and 16 vertices resp.

For this, denote by a' the point where ray $\overrightarrow{x'a}$ hits the boundary of S_x. Consider the reflex vertices of S_x along its boundary in orientation xax' from x to a'. Denote by z the last reflex vertex whose dissecting ray hits the boundary of S on the arc az. The complementary floodlight at z illuminates C. □

Remark 1. Consider a simple polygon P with a dense decomposition L and tree $(G(L), r)$. There is a placement of vertex π-floodlights in P such that every dense

polygon of L with 2ℓ vertices is illuminated by ℓ floodlight (i.e., also there is at most one floodlight at each vertex of P).

In each dense polygon $S \in L$, place a π-floodlight at a non-base outer vertex, and another at most $\ell - 1$ complementary π-floodlights at reflex vertices according to Lemma 3. If a vertex v is a common vertex of two dense sub-polygons S_1 and S_2, and we placed two floodlights at v, then one of them is a complementary floodlight. Hence actually, we place at most one π-floodlight at each vertex.

Such a placement of floodlight is called *basic placement*. If L contains no concave quadrilaterals then a basic placement requires $\lfloor 3(n-2)/4 \rfloor$ floodlights.

4 Proof of Lemma 2

Fix a dense decomposition L of polygon P, and a triangulation of each $S \in L$. We illuminate every $S \in L$ by π-floodlights at vertices of S. Every star-polygon and convex quadrilateral of L is illuminated by *basic* placement described in Remark 1. Every concave quadrilateral of L is illuminated either by two floodlights of a basic placement or by one floodlight. The main concern of our proof is to guarantee that the majority of the concave quadrilaterals require only one floodlight. Then 2ℓ triangles of ℓ concave quadrilaterals require at most $\lfloor 3(2\ell)/4 \rfloor$ floodlights, proving Lemma 2.

A basic placement is not necessarily unique. A star-polygon or a convex quadrilateral $S \in L$ has at least two non-base outer vertices. If $Q \in L$ is illuminated by a floodlight at its non-base outer vertex v, then we color v red. We make our choice using a voting function: A number of concave quadrilaterals vote for each possible non-base outer vertex. We have another choice to make: a floodlight at a non-base outer vertex v can be directed in two different ways. Again a number of concave quadrilaterals vote at each possible non-base outer vertex. The winners require one floodlight, the losers require two.

It is enough to take care of the following two properties. Every concave quadrilateral of L vote at most once. And every concave quadrilateral which does not vote at all requires one floodlight.

4.1 Classifying Convex Quadrilaterals

Let $\mathcal{R} \subset L$ be the set of concave quadrilaterals of L. Denote by $\mathcal{Q}(v) \subset \mathcal{R}$ the set of concave quadrilaterals whose corner vertex is v.

We define recursively two functions g_1 and g_2 on concave quadrilaterals. Suppose that g_1 and g_2 are defined on all descendants of $Q \in \mathcal{R}$. If the reflex vertex of Q is a base vertex then let $g_1(Q) = g_2(Q) = 0$. Assume that $Q = abcd$ where d is non-base reflex vertex, and b is a base corner vertex. Denote by H_d^+ and H_d^- the two halfplanes determined by the line bd such that $a \in H_d^+$ and $c \in H_d^-$. Partition $\mathcal{Q}(d)$ into two sets, $\mathcal{Q}_d^+$ and $\mathcal{Q}_d^-$, such that reflex vertex of all $W \in \mathcal{Q}_d^+$ (resp. $W \in \mathcal{Q}_d^-$) is in H_d^+ (resp. in H_d^-). Let $Q_d(d) \in \mathcal{Q}(d)$ denote the possible quadrilateral dissected by the line bd. Now let

$$g^+(Q) = \mathcal{Q}_d^+ \cup g^-(Q_d) \quad \text{and} \quad \mathrm{g^-(Q) = \mathcal{Q}_d^- \cup g^+(Q_d)}.$$

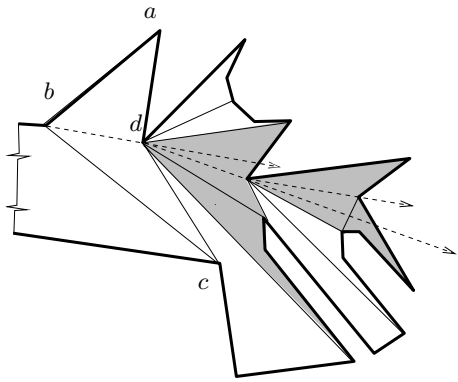


Fig. 3. A dense decomposition of a polygon, where $g_1(abcd)$ is shaded.

We define recursively a function f on non-base outer vertices of dense polygons of L. If f is defined for all non-base outer vertices of all descendant of $S \in L$, then consider a non-base outer vertex v such that u, v, and w are consecutive vertices of S. Let H_1^+ and H_1^- (resp. H_2^+ and H_2^-) be the half-planes determined by uv (resp wv) such that $w \in H_1^+$ (resp. $u \in H_2^+$).

The quadrilaterals in $\mathcal{Q}(v)$ are sorted into tree distinct types. Denote by $\mathcal{Q}_v^A, \mathcal{Q}_v^C$, and $\mathcal{Q}_v^D$ the set of quadrilaterals whose reflex angle is in $H_1 \cap H_2$, $H_1^+ \setminus H_2^-$, and $H_2^+ \setminus H_1^+$ respectively. Let $Q_1(v) \in \mathcal{Q}(v)$ and $Q_2(v) \in \mathcal{Q}(v)$ be the possible quadrilaterals dissected by the line uv and wv resp. (Possibly $Q_1(v) = Q_2(v)$.) See Fig. 4 and 5 for illustrations.

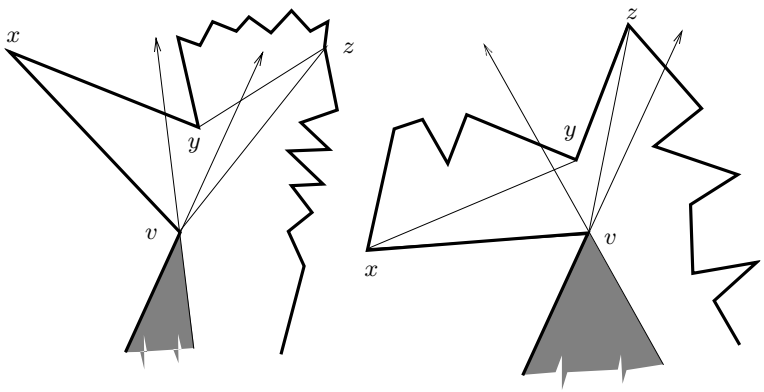


Fig. 4. Polygons where $\alpha(v)$ is shaded and $vxyz$ is of type A. Also they are $Q_1(v) = Q_2(v)$ and $Q_1(v)$ resp.

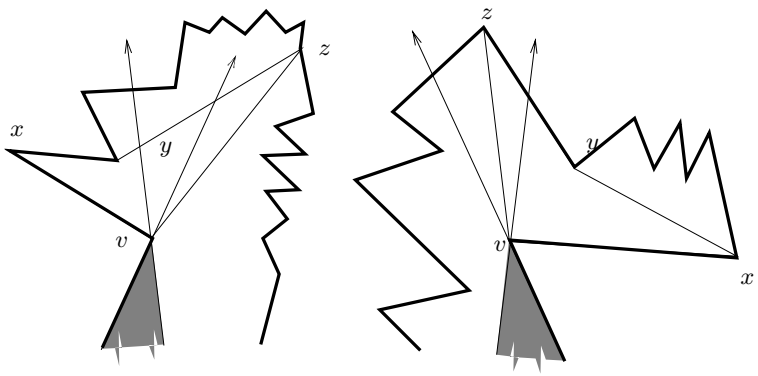


Fig. 5. Polygons where $\alpha(v)$ is shaded and $vxyz$ is of type C, and D resp. Also they are $Q_1(v) = Q_2(v)$ and $Q_2(v)$ resp.

Now let

$$f(v) = \mathcal{Q}_A^v \cup \bigcup_{W \in \mathcal{Q}_A^v} f(W),$$

where $f(W) = f(a)$ for the unique non-base outer vertex a of the concave quadrilateral W. Finally, let

$$h^+(v) = \mathcal{Q}_v^C \cup \bigcup_{W \in \mathcal{Q}_v^C} f(W) \cup g^-(Q_1(v)) \text{ and } \mathrm{h^-(v)} = \mathcal{Q}_\mathrm{v}^\mathrm{D} \cup \bigcup_{\mathrm{W} \in \mathcal{Q}_\mathrm{v}^\mathrm{D}} \mathrm{f(W) \cup g^+(Q_2(v))}.$$

Proposition 4.
(1) $f(v)$, $h^+(v)$, and $h^-(v)$ are disjoint at each vertex v of P.
(2) $f(u) \cup h^+(u) \cup h^-(u)$ and $f(v) \cup h^+(v) \cup h^-(v)$ are disjoint for two non-base vertices of star-polygons u and v.

Proof. Statement (1) is clear from the tree structure of $(G(L), r)$.

For (2), we assign a non-base vertex $r(Q)$ of a star-polygon to each element of $Q \in \mathcal{R}$. For a $Q \in \mathcal{R}$, let $(Q = Q_0, Q_1, \ldots, Q_t)$ be the longest ascending path in $(G(L), r)$ such that $Q_i \in \mathcal{R}$ for all element of the sequence and for all pair (Q_i, Q_{i+1}), the corner vertex of Q_i is a base vertex. (E.g. if the corner vertex of Q is a non-base vertex, then the longest sequence is $(Q = Q_0)$.) Now observe that $Q \in f(v) \cup h^+(v) \cup h^-(v)$ if the corner vertex $r(Q)$ of Q_t is v and v is a non-base vertex of a star-polygon. □

4.2 Placement of Vertex π-Floodlights

The functions f, g and h were defined recursively in ascending order in $G(L)$. The placement of vertex π-floodlights is done in descending order on the tree $(G(L), r)$. We describe a floodlight-placement algorithm. Step I colors red certain

non-base outer vertices of star-polygons and convex quadrilaterals. Step II colors red or blue concave quadrilaterals. We apply a *basic placement* to star polygons with a floodlight at the red vertex and to red quadrilateral. In Step III, each blue concave quadrilateral is illuminated by adding at most one vertex π-floodlight.

Algorithm:

Step I, In each star polygon and convex quadrilateral of L, color a non-base outer vertex v *red*, where $|f(v)|$ is minimal. Color all elements of $f(v)$ red, and all elements of $f(w)$ blue for all other non-base outer vertices.

Step II, For each vertex v of P with $h^+(v) \neq \emptyset$ and $h^-(v) \neq \emptyset$, we make a decision. If $h^+(v) \geq h^-(v)$ (resp. $h^+(v) < h^-(v)$) then color every element of $h^-(v)$ (resp. $h^+(v)$) red and every element of $h^+(v)$ (resp. $h^-(v)$) blue.

Color the non-base outer vertex of each red concave quadrilateral red as well.

Step III, Consider a vertex v with $\mathcal{Q}(v) \neq \emptyset$.

First suppose that $|\mathcal{Q}(v)| = 1$ and v is not red. Place a floodlight at v to illuminate $Q \in \mathcal{Q}(v)$. From now on, we assume $|\mathcal{Q}(v)| > 1$.

Suppose that v is a non-red convex non-base vertex of a dense polygon S, and $h^+(v) \geq h^-(v)$. Place a floodlight at (v, H_2^-). We show that each quadrilateral of $f(v) \cup h^+(v)$ can be illuminated by at most one floodlight. Every $Q \in \mathcal{Q}(v)$ in H_1^+ is illuminated by (v, H_1^+), consider the possible case where $abcd = Q_2(v)$ is in $f(v) \cup h^+(v)$. Triangle $abd \subset abcd$ is illuminated by (v, H_1^+). Place a floodlight at (d, H_d^+) to illuminate triangle abd as well. d is an inner vertex of $abcd$, so basic placements may place at most a complementer floodlight at d.

Suppose that v is a red outer non-base vertex of a dense polygon S and $h^+(v) \geq h^-(v)$. That is, the floodlight at v is assigned to S, and it should illuminate the angular domain uvw. Place a floodlight at (v, H_1^+). We show that each quadrilateral of $h^+(v)$ can be illuminated by at most one additional floodlight. Every $Q \in \mathcal{Q}(v)$ in H_1^+ is illuminated by (v, H_1^+), consider the possible $abcd = Q_2(v) \in \mathcal{Q}(v)$. Triangle $bcd \subset abcd$ is illuminated by (v, H_1^+), place a floodlight at (d, H_d^+) to illuminate triangle abd as well. d is an inner vertex of $abcd$, so basic placements may place at most a complementer floodlight at d.

If v is a reflex vertex of a star-polygon S, then one π-floodlight at v can illuminate every quadrilateral of $\mathcal{Q}(v)$. This is the case also if $v = d$ is a non-base reflex vertex of a concave quadrilateral $abcd$, and there is no floodlight at either (d, H_d^-) nor at (d, H_d^+).

Suppose that $v = d$ is a reflex vertex of a concave quadrilateral $abcd$ and there is a floodlight at, say, (d, H_d^-). It illuminates elements of $\mathcal{Q}_d^+$ except a possible $Q_d(d) \in \mathcal{Q}(d)$, and elements of $\mathcal{Q}_d^-$ are colored red. One triangle of $Q_d(D) = a'b'c'd'$ is illuminated by (d, H_d^-), place a floodlight at $(d', H_{d'}^+)$ to illuminate the other triangle as well. d' is an inner vertex of $a'b'c'd'$, so basic placements may place at most a complementer floodlight at d'.

During the algorithm, we assigned 0, 1, or 2 floodlights to each concave quadrilateral. We assigned 2 floodlights to a concave quadrilateral if and only if it is colored red. The comparisons of $|f(v)|$ and $|h^+(v)|, |h^-(v)|$ guarantee that the majority of concave quadrilaterals are colored blue.

Fig. 6 illustrates the output of our algorithm on a polygon with a dense decomposition where the base side of the root polygon is the upper horizontal segment.

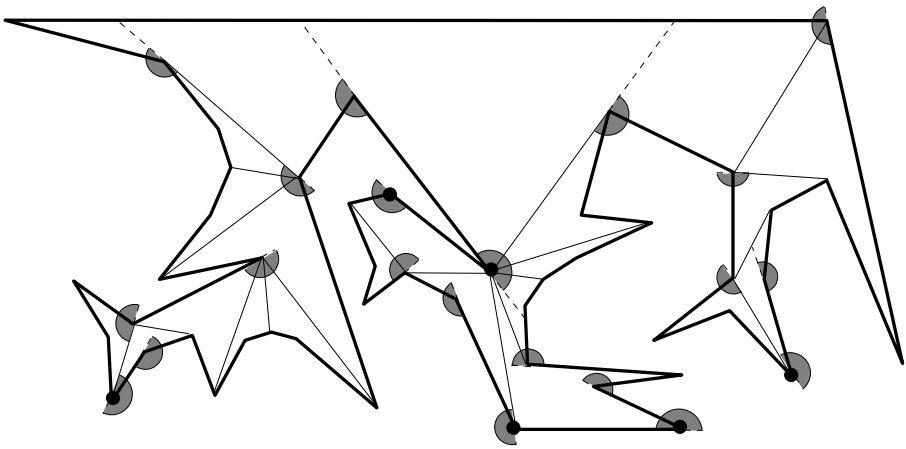


Fig. 6. Placement of floodlights produced by our algorithm on dense decomposition of a polygon.

References

1. Chvátal, V., A combinatorial theorem in plane geometry, *J. Combinatorial Theory Ser. B* **18** (1975), 39–41.
2. Estivill-Castro, V., O'Rourke, J., Urrutia, J., and Xu, D., Illumination of polygons with vertex guards, *Inform. Process. Lett.* **56** (1995) 9–13.
3. Estivill-Castro, V. and Urrutia, J., Optimal floodlight illumination of orthogonal art galleries, in *Proc of the 6th Canad. Conf. Comput. Geom.* (1994) 81–86.
4. Fisk, S., A short proof of Chvátal's watchman theorem, *J. Combinatorial Theory Ser. B* **24** (1978), 374.
5. O'Rourke, J., Open problems in the combinatorics of visibility and illumination, in: *Advances in Discrete and Computational Geometry (B. Chazelle, J. E. Goodman, and R. Pollack, eds.)*, AMS, Providence, 1998, 237–243.
6. O'Rourke J., Art gallery theorems and algorithms, *The International Series of Monographs on Computer Science*, Oxford University Press, New York, 1987.
7. Tóth, Cs. D., Art gallery problem with guards whose range of vision is 180°, *Comput. Geom.* **17** (2000) 121–134.
8. Tóth Cs. D., Floodlight illumination of polygons with uniform 45° angles, submitted.
9. Urrutia, J., Art Gallery and Illumination Problems, in: *Handbook on Computational Geometry (J. R. Sack, J. Urrutia eds.)*, Elsevier, Amsterdam, 2000, 973–1027.

Performance Tradeoffs in Multi-tier Formulation of a Finite Difference Method

Scott B. Baden[1] and Daniel Shalit[1]

University of California, San Diego
Department of Computer Science and Engineering
9500 Gilman Drive, La Jolla, CA 92093-0114 USA
baden@cs.ucsd.edu,dshalit@cs.ucsd.edu
http://www.cse.ucsd.edu/users/{baden,dshalit}

Abstract. *Multi-tier* platforms are hierarchically organized multicomputers with multiprocessor nodes. Compared with previous-generation *single-tier* systems based on uniprocessor nodes, they present a more complex array of performance tradeoffs. We describe performance programming techniques targeted to finite difference methods running on two large scale multi-tier computers manufactured by IBM: NPACI's Blue Horizon and ASCI Blue-Pacific Combined Technology Refresh. Our techniques resulted in performance improvements ranging from 10% to 17% over a traditional single-tier SPMD implementation.

1 Introduction

Multi-tier computers are hierarchically organized multicomputers with enhanced processing nodes built from multiprocessors [13]. They offer the benefit of increased computational capacity while conserving a costly component: the switch. As a result, multi-tier platforms offer potentially unprecedented levels of performance, but increase the opportunity cost of communication [8,1,4]

We have previously described *multi-tier* programming techniques that utilize knowledge of the hierarchical hardware organization to improve performance [2]. These results were obtained on SMP clusters with tens of processors and hence did not demonstrate scalability. In this paper, we extend our techniques to larger-scale multi-tier parallelism involving hundreds of processors, and to deeper memory hierarchies. We describe architecture-cognizant policies needed to deliver high performance in a 3D iterative finite difference method for solving elliptic partial differential equations. 3D Elliptic solvers are particularly challenging owing to their high memory bandwidth requirements. We were able to improve performance over a traditional SPMD implementation by 10% to 17%.

The contribution of this paper is a methodology for realizing overlap on large-scale multi-tier platforms with deep memory hierarchies. We find that uniform partitionings traditionally employed for iterative methods are ineffective, and that irregular, multi-level decompositions are needed instead. Moreover, when reformulating an algorithm to overlap communication with computation, we must avoid even small amounts of load imbalance. These can limit the ability to realize overlap.

V.N. Alexandrov et al. (Eds.): ICCS 2001, LNCS 2073, pp. 785–794, 2001.
© Springer-Verlag Berlin Heidelberg 2001

2 Motivating Application

2.1 A Finite Difference Method

Our motivating application solves a partial differential equation–Poisson's equation in three dimensions. The solver discretizes the equation using a 7-point stencil, and solves the discrete equation on a 3-d mesh using Gauss-Seidel's method with red-black ordering. We will refer to this application as *RedBlack3D*.

We assume a hierarchically constructed multicomputer with N processing nodes. Each node is a shared memory multiprocessor with p processors. When $p = 1$ our machine reduces the degenerate case of a *single-tier* computer with a flattened communication structure. For $p > 1$ we have a *multi-tier* computer.

Our strategy for parallelizing an iterative method is to employ a blocked hierarchical decomposition, reflecting the hierarchical construction of the hardware [1,2]. Fig. 1 shows hierarchical decomposition. The first-level subdivision (Fig. 1a) splits the computational domain into N uniform, disjoint blocks or subdomains. The second level (Fig. 1b) subdivides each of the N blocks into p disjoint sub-blocks. Each first-level block is buffered by a surrounding *ghost region* holding off-processor values.

The calculation consists of successive steps that compute and then communicate to fill the ghost cells. After communication of ghost cells completes, control flow proceeds in hierarchical fashion, passing successively to node-level and then processor-level execution.

Each node sweeps over its assigned mesh, enabling its processors to execute over a unique sub-block. Once the processors finish computing, control flow lifts back up to the node level: each node synchronizes its processors at a barrier, and the cycle repeats until convergence.

Under this hierarchical model, nodes communicate by passing messages on behalf of their processors. Since ghost cells are associated with nodes rather than processors, processors on different nodes do not communicate directly.

2.2 Overlap

Communication delays are long on a multi-tier computer because multiple processors share a communication port to the interconnection network. To cope with long communication delays, we reformulate the iterative method to overlap communication with computation by pre-fetching the ghost cells [14].

As illustrated in Fig. 1(b), we peel an annular region from surface of each node's assigned subdomain, and defer execution on this annulus until the ghost cells have arrived. We initiate communication asynchronously on the ghost cells, and then compute on the interior of the subdomain, excluding the annular region. This is shown in Fig. 1(b). After computation finishes, we wait for communication to complete. Finally, we compute over the annular region.

We now have the basis for building an efficient iterative method on a multi-tier computer. We next discuss the performance programming techniques required to implement the strategy.

Fig. 1. (a) Cross section of a 3D problem partitioned across 4 nodes, showing the halo region; and (b) the node-level partitioning on dual-processor nodes. The halo is shaded in this depiction. The annular region abuts the halo, and is subdivided into pieces labeled 2 through 5. Points on the interior are labeled 0 and 1. This decomposition is duplicated on each node.

3 Testbeds

3.1 Hardware

We ran on two platforms, both developed by IBM: NPACI's Blue Horizon system[1], located at the San Diego Supercomputer Center, and the ASCI Blue Pacific Combined Technology Refresh (CTR)[2], located at Lawrence Livermore National Laboratory. The two platforms differ significantly in their respective on-node memory hierarchies. Blue Horizon provides significantly lower node bisection bandwidth than CTR relative to processor performance. The nodes are over an order of magnitude more powerful and have twice the number of processors. Blue Horizon's shared memory is multi-ported and employs a cross-bar interconnect rather than a bus. The cache lines are longer.

Blue Horizon contains 144 POWER3 SMP High Nodes (model number 9076-260) interconnected with a "Colony" switch. Each node is an 8-way way Symmetric Multiprocessor (SMP) based on 375 MHz Power-3 processors, sharing 4 Gigabytes of memory, and running AIX 4.3. Each processor has 1.5 GB/sec bandwidth to memory, an 8 MB 4-way set associative L2 cache, and 64 KB of 128-way set associative L1 cache. Both caches have a 128 byte line size.

Blue Pacific contains 320 nodes. Each node is a model number 9076-WCN 4-way SMP based on 332 MHz Power PC 604e processors sharing 1.5 GB memory and running AIX 4.3.1. Each processor has 1.33 GB/sec of bandwidth to memory, a 32 KB 4-way set associative L1 data cache with a 32 byte line size, and a 256KB direct-mapped, unified L2 cache with a 64 byte line size.

We used KAI's C++ and Fortran 77 compilers. These compilers are translators, and employ native IBM compilers to generate object code. C++ code was compiled `kai_mpCC_r`, with compiler options `--exceptions -O2`

[1] `http://www.npaci.edu/BlueHorizon/`
[2] `http://www.llnl.gov/asci/platforms/bluepac/`

`-qmaxmem=-1 -qarch=auto -qtune=auto --no_implicit_include`. Fortran 77 was compiled using `guidef77`, version 3.9, with compiler options `-O3 -qstrict -u -qarch=pwr3 -qtune=pwr3`.[3]

3.2 Performance Measurement Technique

We collected timings in batch mode: *Distributed Production Control System (DPCS)* on ASCI Blue Pacific, *loadleveler* on NPACI Blue Horizon. We report wall-clock times obtained with `read_real_time()` on Blue Pacific, and `MPI_Wtime()` on Blue Horizon.

The timed computation was repeated for a sufficient number of iterations to ensure that the entire run lasted for tens of seconds. Times were reported as the average of 20 runs, with occasional outliers removed. We define an outlier as running at least 25% more slowly than the average time of the other runs. In practice, we encountered outliers once or twice in each batch of twenty runs.

3.3 KeLP Software Testbed

The applications were written in a mix of C++ and Fortran 77 and used a multi-tier prototype of the KeLP infrastructure [1,2,4]. KeLP calls were made from C++, and all numerical computation was carried out in Fortran. A discussion of the KeLP API is out of the scope of this paper. The interested reader is referred to the above references for more information.

KeLP employs POSIX threads [7] to manage parallelism on node, and MPI [6] to handle communication between nodes. A typical KeLP program runs with one MPI process per node, and unfolds a user-selectable number of threads within each process. The total number of threads per node is generally equal to the number of processors.

KeLP employs a persistent communication object called a *Mover* [5] to move data between nodes. A distinguished master thread in each process is in charge of invoking the Mover, which logically runs as a separate task. Mover provides two entries for managing communication asynchronously: *start()* and *wait()*.

KeLP provides two implementation policies for supporting asynchronous, non-blocking communication in the Mover. The Mover may either run as a proxy [12] within a separate thread, or it may be invoked directly by the master thread. In the latter case, asynchronous non-blocking MPI calls `MPI_Isend()` and `MPI_Irecv()` are relied on to provide overlap. However, we found that IBM's MPI implementation cannot realize communication overlap non-blocking asynchronous communication. Thus, we use only the proxy to realize overlap.

4 Results

4.1 Variant Policies

We implemented several variant policies, which are summarized in Table 1. The simplest variant, HAND, is hand-coded in MPI. This variant is typical of how

[3] On Blue Pacific we compiled with options `-qarch=auto -qtune=auto` in lieu of `pwr3`.

most users would implement RedBlack3D, and execution is single-tiered. All other variants were written in KeLP, and used the identical numerical Fortran 77 kernel.

Table 1. A synopsis of the policy variants used in the paper.

Variant	Explanation
HAND	Hand coded MPI (single-tier)
MT(K)	$k > 1$: Multi-tier implementation with k threads per process
MT(1)	Single-tier implementation written in KeLP
OLAP	Communication overlap
!OLAP	No communication overlap
XTRA	Use an extra thread (proxy) to overlap communication
IRR(M,N)	Irregular decomposition assigning work to two processor sets in the ratio m: n

The next variant is MT(P). It supports multi-tier execution using p computation threads per node. With $p = 1$, we flatten out the hierarchical machine interconnection structure. Thus, MT(1) reduces to single-tier execution, running 1 process per processor. When $p > 1$, we obtain a family of multi-tier variants.

We compose the overlap variant with MT(P). As discussed previously, we use a proxy to overlap communication with computation. To signify this overlap variant, we concatenate the policy XTRA using the + sign to indicate variant concatenation. Thus, the policy MT(P)+OLAP+XTRA employs multi-tier execution with p compute threads, and supports communication overlap using an extra thread running a proxy. We will use the variant !OLAP to indicate when we do not employ overlap.

4.2 Experimentation

We first present results for Blue Pacific CTR and then for Blue Horizon. We report all performance figures as the average number of milliseconds per iteration, and ran for 80 iterations. As noted previously, we report the average of 20 runs, ignoring outliers. On Blue Pacific CTR, we ran with a 480^3 domain on 64 nodes (256 processors). On Blue Horizon, we ran with 8 and 27 nodes (64 and 216 processors, respectively), keeping the problem size constant with the number of processors.

Establishing a Baseline. To establish the operating overheads of KeLP, we compare HAND against MT(1)+!OLAP. An iteration of MT(1)+!OLAP completes in 245 ms., including 116 ms of communication wait time. By comparison, HAND completes in 229 ms., including 99.3 ms of communication wait time. KeLP overheads are modest and incurred primarily in communication (15%).

Overall, the application runs just 7% more slowly in KeLP than in MPI. Having determined that KeLP's overheads are low, we will use the single-tier variant written in KeLP, MT(1)+!Olap, as our baseline for assessing the benefits of multi-tier execution.

Multi-tier execution. We next run with MT(p) using Olap and !Olap variants.[4] To peform these runs, we employed the following AIX environment variable settings: `MP_SINGLE_THREAD=yes;AIXTHREAD_SCOPE=S`. Additionally, the Olap variant ran with `MP_CSS_INTERRUPT=yes`. The !Olap variant ran with
`MP_CSS_INTERRUPT=no; MP_POLLING_INTERVAL=2000000000`.

Compared with MT(1), MT(4)+ !Olap reduces the running time slightly from 245 ms to 234 ms. Computation time is virtually unchanged. Communication time drops about 15%. We attribute the difference to the use of the shared memory cache-coherence protocol to manage interprocessor communication in lieu of message passing. Although Blue Pacific uses shared memory to resolve message passing on-node, communication bandwidth is about 80 Megabytes/sec regardless of whether or not the communicating processors are on the same node. As noted previously, bandwidth to memory is more than an order of magnitude higher: 1.33 GB/sec per processor. We are now running at about the same speed as hand-coded MPI. Our next variant will improve performance beyond the HAND variant.

Overlap. We next ran MT(3)+Olap+Xtra. Performance improves by about 11% over MT(4)+ !Olap: execution time drops to 209 ms. We are now running 17% faster than the single-tier variant. Communication wait time drops to 29.6 ms–a reduction of a factor of three. The proxy is doing its job, overlapping most of the communication. Since the proxy displaces one computational thread, we expect an increase in computation time. Indeed, computation time increases from 139 ms to 184 ms. This slowdown forms the ratio of 3:4, which is precisely the increase in workload that results results from displacing one computational thread by the proxy.

Although communication wait time has dropped significantly, it is still nonzero. Proxy utilization is only about 25% so this is not at issue. Part of the loss results from thread synchronization overhead. But load imbalance is also a significant factor. It arises in the computation over the inner annular region. The annulus is divided into six faces, and each face is assigned to one thread. (Faces that abut a physical boundary have 3, 4, or 5 faces.) Because faces have different strides–depending on their spatial orientation– the computation over the annulus completes at different times on different nodes. The resulting imbalances delay communication at the start of the next iteration. The time lag compounds over successive iterations, causing a phase shift in communication. When this phase shift is sufficiently long, there is not sufficient time for com-

[4] We did not run MT(1)+Olap since the p extra proxy threads would interfere uselessly with one another.

munication to complete prior to the end of computation. We estimate that this phase shift accounts for 1/3 to 1/2 of the total wait time.

Tab. 2 summarizes performance of variants of HAND, MT(1)+!Olap, MT(4)+!Olap, and MT(3)+Olap+Xtra.

Table 2. Execution time break-down for variants of redblack3D running on 64 nodes of ASCI Blue Pacific CTR. Times are reported in milliseconds per iteration. The column labeled 'Wait' reports the time spent waiting for communication to complete. The times reported are the maximum reported from all nodes; thus, the local computation and communication times do not add up exactly to the total time.

Variant	Total	Wait	Comp
HAND	229	99.3	147
MT(1) + !Olap	245	116	142
MT(4) + !Olap	234	100	139
MT(3) + Olap + Xtra	209	29.6	184

Blue Horizon. Blue Horizon has has a "Colony switch," that provides about 400 MB/sec of message bandwidth under MPI for off-node communication, and 500 MB/sec on-node. We used AIX environment variables recommended by SDSC and IBM. For non-overlapped runs we used
`#@ Environment = COPY_ALL; MP_EUILIB=us; MP_PULSE=0; MP_CPU_USAGE=unique; MP_SHARED_MEMORY=YES; AIXTHREAD_SCOPE=S; RT_GRQ=ON; MP_INTRDELAY=100;` for overlapped runs we added the settings `MP_POLLING_INTERVAL=2000000000; AIXTHREAD_MNRATIO=8:8`. With single-tier runs, the load leveler variable `tasks_per_node=8`. For MT(P) , $p > 1$, we used a value of 1. The number of nodes equals the number of MPI processes.

We ran on 8 and 27 nodes, 64 and 216 processors, respectively. We maintained a constant workload per node, running with a 800^3 mesh on 8 nodes, and a 1200^3 mesh 27 nodes. This problem size was chosen to utilize 1/4 of the nodes' 4GB of memory. In practice, we would have many more than the 2 arrays used in RedBlack3D (solution and right hand side), and would not likely be able to run with a larger value of N. Tab. 3 summarizes performance.

We first verify that KeLP overheads are small. Indeed, the KeLP (MT(1)+!Olap) and Hand variants run in nearly the identical amount of time. The multi-tier variant MT(8)+ !Olap reduces the running time from 732 ms to 713 ms on 8 nodes. Curiously the running time *increases* on 27 nodes, from 773 ms to 824 ms. The increase is in communication time–computation time is virtually unchanged. Possibly, external communication interference increases with a larger number of nodes, and is affecting communication performance. We are currently investigating this effect.

The benefits of multi-tier parallelism come with the next variant: communication overlap. MT(7) + Olap runs faster than MT(1) + !Olap, reducing

Table 3. Execution time break-down for variants of redblack3D running on 8 and 64 nodes of NPACI Blue Horizon, with N=800 and 1200, respectively. The legend is the same as the previous table. Threads were unbound except for MT(7) + OLAP + XTRA + IRR(44:50). We were unable to run the HAND variant on 8 nodes due to a limitation in the code. We were unable to get speedups in the IRR variant on 27 nodes.

Variant	Total	Wait	Comp	Total	Wait	Comp
HAND	719	144	575	-	-	-
MT(1)+!OLAP	732	163	569	773	195	578
MT(8)+!OLAP	713	141	566	824	230	581
MT(7)+OLAP+XTRA	655	20.5	608	693	42.5	669
MT(7)+OLAP+XTRA+IRR(44:50)	626	9.23	607	-	-	-

execution time to 655 ms on 8 nodes, and 693 on 27 nodes. Overlap significantly reduces the wait time on communication, which drops from 141 ms to 20.5 ms on 8 nodes, and from 230 ms to 42.5 ms on 27 nodes. Our multi-tier overlapped variant MT(7) + OLAP is about 10% faster than the single-tier variant MT(1) + !OLAP. Although our strategy increases computation time, more significantly, it reduces the length of the critical path: communication.

An additional level of the memory hierarchy. Although we have reduced communication time significantly, there is still room for improvement. Upon closer examination, the workload carried by the computational threads on the interior of the domain is imbalanced. This imbalance is in addition to the imbalance within the annulus, which was discussed above.

The reason why is that the Power3 high node's shared memory is organized into groups of four processors and each group has one port to memory. Thus, when we run with seven compute threads, four of the threads sharing one port of memory see less per-CPU bandwidth than the other three threads sharing the other port. The uniform partitionings we used are designed to divide floating point operations evenly, but not memory bandwidth requirements.

The thread scheduler does a good job of mitigating the load imbalance, but at a cost of increased overheads. We can reduce running time further by explicitly load balancing the threads' workload assignments according the available per-processor bandwith. We use an irregular hierarchical partitioning. The first level divides the inner computational domain into two parts, such that the relative sizes of the two parts correspond to an equal amount of bandwidth per processor. We determined experimentally that a ratio of 44:50 worked best. That is, 44/94 of the 504 planes in the domain were assigned contiguously to 4 processors, and the remainder to the other 3 processors.

The irregular hierarchical improve performance, cutting the communication wait time in half to 9.2 ms. Overall computation time drops to 626 ms. We have now improved performance by 14.4% relative to the single-tier implementation.

As with ASCI Blue, it appears that the remaining losses result from thread synchronization overheads and from load imbalances arising within the annulus computation. The latter effect is more severe on Blue Horizon, which has 8 way nodes, than with Blue Pacific CTR, which has only 4-way nodes. To avoid large memory access strides in the annulus computation, we were limited to two-dimensional data decompositions. (Long strides, comprising thousands of bytes, penalize computation severely on unfavorably oriented faces–by a factor of 20!) No node received more than 4 annular faces. We can only utilize about half the 7 processors on Blue Horizon when computing on the annulus. The load imbalance due to the annulus computation introduces a phase lag of about 3% into the iteration cycle. Communication within the proxy consumes about 18%. Thus, after about 25 iterations, we can no longer overlap communication. Our runs were 40 cycles long.

5 Conclusions and Related Work

We have presented a set of performance programming techniques that are capable of reducing communication delays significantly on multi-tier architectures that employ a hierarchical organization using multiprocessor nodes. We realized improvements in the range of 10% to 17% for a 3D elliptic solver. A drawback of our approach–and others that employ *hybrid* programming–is to introduce a more complicated hierarchical programming model and a more complicated set of performance tradeoffs. This model has a steeper learning curve than traditional SPMD programming models, but is appropriate when performance is at a premium. Our data decompositions were highly irregular, and we were constantly fighting load imbalance problems. We suspect that dynamic workload sharing on the node would be easier to program and more effective in dealing with the wide range of architectural choices faced by users of multi-tier systems.

Other have incorporated hierarchical abstractions into programming languages. Crandall *et. al* [10] report experiences with dual-level parallel programs on an SMP cluster. Cedar Fortran [9] included storage classes and looping constructs to express multiple levels of parallelism and locality for the Cedar machine. The pSather language is based on a cluster machine model for specifying locality [11], and implements a two-level shared address space.

Acknowledgments

The authors wish to thank John May and Bronis de Supinski, with the Center for Applied Scientific Computing at Lawrence Livermore National Laboratory, for the many illuminating discussions about the ASCI Blue-Pacific machine and with Dr. Bill Tuel, and David Klepacki, both with IBM, for explaining the subtleties of performance tuning in IBM SP systems.

KeLP was the thesis topic of Stephen J. Fink (Ph.D. 1998), who was supported by the DOE Computational Science Graduate Fellowship Program. Scott Baden is supported in part by NSF contract ACI-9876923 and in part by NSF

contract ACI-9619020, "National Partnership for Advanced Computational Infrastructure." Work on the ASCI Blue-Pacific CTR machine was performed under the auspices of the US Dept of Energy by Lawrence Livermore National Laboratory Under Contract W07405-Eng-48.

References

1. Fink, S. J.: Hierarchical Programming for Block–Structured Scientific Calculations. Doctor dissertation, Dept. of Computer Science and Engineering, Univ. of Calif., San Diego (1998)
2. Baden, S.B. and Fink, S. J.: Communication Overlap in Multi-tier Parallel Algorithms. In Proc. SC '98, IEEE Computer Society Press (1998)
3. Fink, S. J. and Baden, S.B. Runtime Support for Multi-tier Programming of Block-Structured Applications on SMP Clusters. In: Ishikawa, Y., Oldehoeft, R, Reynders, J.V.W., and Tholburn, M. (eds.): Scientific Computing in Object-Oriented Parallel Environments. Lecture Notes in Computer Sci., Vol. 1343. Springer-Verlag, New York (1997) pp. 1–8
4. Fink, S. J. and Baden, S.B. A Programming Methodology for Dual-tier Multicomputers. *IEEE Trans. on Software Eng.,* 26(3), March 2000, pp. 212–26
5. Baden, S.B. and Fink, S. J., and Kohn, S. R. Efficient Run-Time Support for Irregular Block-Structured Applications. *J. Parallel Distrib. Comput.,* Vol 50, 1998, pp. 61–82
6. MPI Forum: The Message Passing Interface (MPI) Standard. `http://www-unix.mcs.anl.gov/mpi/index.html`, 1995
7. IEEE: IEEE Guide to the POSIX Open System Environment. New York, NY, 1995
8. Gropp, W.W. and Lusk, E. L. A Taxonomy of Programming Models for Symmetric Multiprocessors and SMP Clusters. In Giloi, W. K. and Jahnichen, S., and Shriver, B. D. (eds.): Programming Models for Massively Parallel Computers. IEEE Computer Society Press, 1995, pp. 2–7
9. Eigenmann, R., Hoeflinger, J., Jaxson,G., and Padua, D. Cedar Fortran and its Compiler, *CONPAR 90-VAPP IV, Joint Int. Conf. on Vector and Parallel Proc.*, 1990, pp. 288–299
10. Crandall, P. E., Sumithasri, E. V., Leichtl, J., and Clement, M. A. A Taxonomy for Dual-Level Parallelism in Cluster Computing, Tech. Rep., Univ. Connecticut, Mansfield, Dept. Computer Science and Engineering, 1998
11. Murer, S., Feldman, J., Lim, C.-C., and Seidel, M.-M. pSather: Layered Extensions to an Object-Oriented Language for Efficient Parallel Computation, Tech. Rep. TR-93-028, Computer Sci. Div., U.C. Berkeley, Dec. 1993
12. Lim, B.-H., Heidelberger, P., Pattnaik, P., and Snir, M. Message Proxies for Efficient, Protected Communication on SMP Clusters, in Proc. Third Int'l Symp. on High-Performance Computer Architecture, San Antonio, TX, Feb. 1997, IEEE Computer Society Press, pp. 116–27.
13. Woodward, P.R.. Perspectives on Supercomputing: Three Decades of Change, *IEEE Computer*, Vol. 29, Oct. 1996, pp. 99–111.
14. Sawdey, A. C., O'Keefe, M.T., and Jones, W.B. A General Programming Model for Developing Scalable Ocean Circulation Applications, *Proc. ECMWF Workshop on the Use of Parallel Processors in Meteorology*, Jan. 1997.
15. Somani, A. K. and Sansano, A. M. *Minimizing Overhead in Parallel Algorithms through Overlapping Communication/Computation,* Tech. Rep. 97-8, NASA ICASE, Langley, VA., Feb. 1997

On the Use of a Differentiated Finite Element Package for Sensitivity Analysis*

Christian H. Bischof, H. Martin Bücker, Bruno Lang, Arno Rasch, and Jakob W. Risch

Institute for Scientific Computing, Aachen University of Technology,
D–52056 Aachen, Germany
{bischof, buecker, lang, rasch, risch}@sc.rwth-aachen.de
http://www.sc.rwth-aachen.de

Abstract. Derivatives are ubiquitous in various areas of computational science including sensitivity analysis and parameter optimization of computer models. Among the various methods for obtaining derivatives, automatic differentiation (AD) combines freedom from approximation errors, high performance, and the ability to handle arbitrarily complex codes arising from large-scale scientific investigations. In this note, we show how AD technology can aid in the sensitivity analysis of a computer model by considering a classic fluid flow experiment as an example. To this end, the software tool ADIFOR implementing the AD technology for functions written in Fortran 77 was applied to the large finite element package SEPRAN. Differentiated versions of SEPRAN enable sensitivity analysis for a wide range of applications, not only from computational fluid dynamics.

1 Introduction

In assessing the robustness of a computer code, or to determine profitable avenues for improving a design, it is important to know the rate of change of the model output that is implied by changing certain model inputs. Derivatives are one way to implement such a sensitivity analysis. Traditionally, divided differences are employed in this context to approximate derivatives, leading to results of dubious quality at often great computational expense. Automatic differentiation (AD), in contrast, is an alternative for the evaluation of derivatives providing guaranteed accuracy, ease of use, and computational efficiency. Note that derivatives play a crucial role not only in sensitivity analysis but in numerical computing in general. Examples include the solution of nonlinear systems of equations, stiff ordinary differential equations, partial differential equations, differential-algebraic equations, and multidisciplinary design optimization, to name just a few. Therefore, the availability of accurate and efficient derivatives is often indispensable in computational science.

* This research is partially supported by the Deutsche Forschungsgemeinschaft (DFG) within SFB 540 "Model-based experimental analysis of kinetic phenomena in fluid multi-phase reactive systems," Aachen University of Technology, Germany.

V.N. Alexandrov et al. (Eds.): ICCS 2001, LNCS 2073, pp. 795–801, 2001.
© Springer-Verlag Berlin Heidelberg 2001

In this note we give an answer to the following question. Given an arbitrarily complicated computer program in a high-level programming language such as Fortran, C, or C++, how do we get accurate and efficient derivatives for the function implemented by the computer program? We will argue that the answer is to apply automatic differentiation. Although AD is a general technique applicable to programs written in virtually any high-level programming language [1, 4,5,6], we will assume in this note that the function for which derivatives are desired is written in Fortran 77, as it is the case for the package SEPRAN [8]. Developed at "Ingenieursbureau SEPRA" and Delft University of Technology, SEPRAN is a large general purpose finite element code intended to be used for the numerical solution of second order elliptic and parabolic partial differential equations in two and three dimensions. It is employed in a wide variety of engineering applications [3,9,10,11,12,13,14] including structural mechanics and laminar or turbulent flow of incompressible liquids.

In Sect. 2, we describe the basic principles behind the AD technology as well as the application of an AD tool to SEPRAN leading to a differentiated version of SEPRAN called SEPRAN.AD hereafter. The simulation of a classic fluid flow experiment, namely the flow over a 2D backward facing step, is taken as a simple, yet illustrative, example for carrying out numerical experiments in Sect. 3. We show how a SEPRAN user benefits from the preprocessed code SEPRAN.AD in that it provides—with no more effort than is required to run SEPRAN itself—a set of derivatives that is accurate and consistent with the numerical simulation. Finally, we point out that the functionality contained in differentiated versions of SEPRAN allows the sensitivity analysis of a wide range of potential SEPRAN applications, not only from computational fluid dynamics.

2 Automatic Differentiation and SEPRAN

Automatic differentiation is a powerful technique for accurately evaluating derivatives of functions given in the form of a high-level programming language, e.g., Fortran, C, or C++. The reader is referred to the recent book by Griewank [5] and the proceedings of AD workshops [1,4,6] for details on this technique. In automatic differentiation the program is treated as a—potentially very long—sequence of elementary statements such as binary addition or multiplication, for which the derivatives are known. Then the chain rule of differential calculus is applied over and over again, combining these step-wise derivatives to yield the derivatives of the whole program. This mechanical process can be automated, and several AD tools are available that augment a given code C to a new code C.AD such that, in addition to the original outputs, C.AD also computes the derivatives of some of these output variables with respect to selected inputs. This way AD requires little human effort and produces derivatives that are accurate up to machine precision.

The AD technology is not only applicable for small codes but scales up to large codes with several hundreds of thousand lines; see the above-mentioned proceedings and the references given therein. We applied automatic differentiation

to the general purpose finite element package SEPRAN consisting of approximately 400,000 lines of Fortran 77. The package enables simulation in various scientific areas ranging from fluid dynamics, structural mechanics to electromagnetism. Analyses of two-dimensional, axisymmetric and three-dimensional steady state or transient simulations in complex geometries are supported. Examples include potential problems, convection-diffusion problems, Helmholtz-type equations, heat equations, and Navier-Stokes equations.

We used the ADIFOR tool [2] to generate SEPRAN.AD, the differentiated version. ADIFOR (**A**utomatic **DI**fferentiation of **FOR**tran) implements the AD technology for Fortran 77 codes. The details of this process will be presented elsewhere. In general, a user of an AD tool needs to perform the following steps:

1. As a preprocessing step, "dirty" legacy code needs certain manual massaging to produce "clean" code conforming to the language standard. Notice that SEPRAN is programmed in an almost clean way so that only small changes to the original code had to be done by hand, examples being several instances where different routines interpret the same memory as holding either double precision real data or single precision complex data. This non-standard technique is sometimes employed in order to save memory, and it is not detected by current Fortran compilers because their view of the program is restricted to one routine or file at a time. ADIFOR, by contrast, does a global data flow analysis and immediately detects this kind of inconsistency.
2. The user indicates the desired derivatives by specifying the dependent (output) and independent (input) variables. This is typically done through a control file.
3. The tool is then applied to the clean code to produce augmented code for the additional computation of derivatives. We applied ADIFOR 2.1 to SEPRAN (approximately 400,000 lines of code including comments) to obtain SEPRAN.AD (roughly 600,000 lines of code including comments). Note that the global analysis enables ADIFOR to decide whether the work done in a routine is relevant to the desired derivative values. Therefore only a subset of the routines is actually augmented.
4. A small piece of code (driver code) is constructed that calls the generated routines made available by SEPRAN.AD.
5. The generated derivative code and the driver code is compiled and linked with supporting libraries.

Upon successful completion of these steps, derivatives are available by simply calling the corresponding routines from SEPRAN.AD, the differentiated version, rather than from SEPRAN, the original code.

Once the differentiated code is available, it enables sensitivity analysis of different problems (e.g., flow around obstacles, flow over a backward facing step, etc.) with respect to the specified input and output variables. If other variables are to be considered then steps 2 through 5 of the above procedure are repeated, which requires only little human interaction. (There is a slightly more sophisticated way to do it, which even avoids repeating steps 2 and 3.) Note that step 1 is the only step that might need substantial human effort and is done only once.

The above discussion demonstrates the ease of use and the versatility of the AD technology.

3 Results

In the numerical experiments reported in this section, a simulation of a classic fluid flow experiment, namely the flow over a 2D backward facing step [7], is taken as a sample problem. The goal of this note is not to concentrate on the values of the flow field but to give the reader an impression of the improved functionality of the differentiated version SEPRAN.AD as compared to SEPRAN. In this standard benchmark problem for incompressible fluids, a stationary flow over a backward facing step is considered. We carried out numerical experiments at Reynolds numbers around 50 with no-slip boundary conditions at the upper and lower walls of the pipe, a parabolic inflow in horizontal direction, and a parallel outflow.

Given the maximal horizontal velocity component v_0 of the inflow, the density ρ, and the viscosity μ, one can easily use SEPRAN to compute the velocity v and the pressure p at any point in the pipe. From an abstract point of view, the corresponding code implements a function f taking v_0, ρ, and μ as input and producing the output v and p; that is

$$\begin{pmatrix} v \\ p \end{pmatrix} = f(v_0, \rho, \mu) \ .$$

Invoking the corresponding SEPRAN code evaluates f at a given input.

Suppose that we are interested in evaluating the derivatives of some outputs of f with respect to some of its inputs at the same point where f itself is evaluated. For instance, an engineer might be interested in the rate of change of the pressure p with respect to the inflow velocity v_0, i.e., $\partial p/\partial v_0$. A numerical approach would make use of divided differences to approximate the derivative. For the sake of simplicity, we only consider first-order forward divided differences such as

$$\frac{\partial p(v_0, \rho, \mu)}{\partial v_0} \approx \frac{p(v_0 + h, \rho, \mu) - p(v_0, \rho, \mu)}{h} \ , \tag{1}$$

where h is a suitably chosen step size. An advantage of the divided difference approach is its simplicity; that is, the corresponding function is evaluated in a black-box fashion. The main disadvantage of divided differences is that the accuracy of the approximation depends crucially on a suitable step size h. Unfortunately, an optimal or even near-optimal step size is often not known a priori. Therefore, the program is usually run several times to find a reasonable step size. Note that there is a complementary influence of truncation and cancellation error to the overall accuracy of the method: on the one hand, the step size should be as small as possible to decrease the approximation error that would be present even if infinite-precision arithmetic were to be used. On the other hand, the step size must not be too small to avoid cancellation of significant digits when using finite-precision arithmetic in the evaluation of (1).

The above problem of determining a step size is a conceptual disadvantage in the divided difference approach and also applies to higher-order derivatives. Automatic differentiation, on the contrary, does not involve any truncation error. Derivatives produced by AD are exact up to machine precision. To demonstrate the difference in accuracy between AD and divided differences, we formally define

$$\mathrm{diff}(p, v_0) := \left\| \frac{\partial p(v_0, \rho, \mu)}{\partial v_0} - \frac{p(v_0 + h, \rho, \mu) - p(v_0, \rho, \mu)}{h} \right\|_\infty , \qquad (2)$$

where the first term on the right-hand side is the value computed by automatic differentiation. Hence, $\mathrm{diff}(p, v_0)$ is a measure of the difference of the numerical accuracy of the derivatives of p with respect to v_0 obtained from automatic differentiation and divided differences.

For the backward facing step example, the difference between the derivative values generated by AD and divided differences using varying step sizes h is shown in Tab. 1.

Table 1. Comparison of the accuracy of derivatives obtained from divided differences using a step size h and automatic differentiation.

h	$\mathrm{diff}(v, v_0)$	$\mathrm{diff}(v, \rho)$	$\mathrm{diff}(v, \mu)$	$\mathrm{diff}(p, v_0)$	$\mathrm{diff}(p, \rho)$	$\mathrm{diff}(p, \mu)$
10^{-2}	0.002189	0.001134	11.774571	0.002310	0.001087	5.259458
10^{-3}	0.000218	0.000111	2.039314	0.000230	0.000107	0.996281
10^{-4}	0.000043	0.000042	0.217868	0.000032	0.000028	0.107945
10^{-5}	0.000277	0.000251	0.021579	0.000304	0.000326	0.010897
10^{-6}	0.002078	0.003096	0.002766	0.002146	0.001811	0.002294
10^{-7}	0.029861	0.038406	0.027861	0.020655	0.023987	0.028521
10^{-8}	0.197591	0.260977	0.213695	0.155814	0.193424	0.184808
10^{-9}	5.313513	3.374881	3.746390	1.622882	2.115335	3.093727
10^{-10}	25.566379	20.481873	27.184625	21.904384	14.420520	24.604476

Here, the definition (2) is extended to derivatives other than $\partial p/\partial v_0$ in a straight forward fashion. The derivatives of the pressure and the velocity fields are evaluated at $(v_0, \rho, \mu) = (1.0, 1.0, 0.01)$. The table demonstrates the dependence of the divided difference approach from the step size. In all columns of the table, the difference values first decrease with decreasing step size and then increase again, and the optimum step size depends on the particular derivative. For instance, $\mathrm{diff}(p, v_0)$ is minimal for $h = 10^{-4}$ whereas the minimum of $\mathrm{diff}(p, \mu)$ is at $h = 10^{-6}$ indicating the need for finding different suitable step sizes when differentiating with respect to v_0 and μ.

In contrast to divided differences, there is no need for experimenting with step sizes at all when applying automatic differentiation because there is no truncation error. Using AD, the accurate derivative values of p and v with respect to all three input parameters, together with the function values, were obtained

with a single call to the differentiated version SEPRAN.AD. This computation required roughly 3.3 seconds and 95 MB of memory, compared to 1.2 seconds and 26 MB for one run of SEPRAN. Note that using divided differences for approximating the derivatives with respect to three variables requires at least a total of four SEPRAN calls. Thus AD, in addition to providing more reliable results, also takes less time than divided differences. We finally mention that SEPRAN.AD needs additional memory to store the three derivatives. So, the above mentioned increase of a factor of 3.7 is moderate.

4 Concluding Remarks

The technique of automatic differentiation is proved to be an efficient way to obtain accurate derivatives of functions given in the form of a computer program written in any high-level language such as Fortran, C, or C++. The technique scales up to large simulation codes that are used today as a crucial part in a broad variety of scientific and engineering investigations. We applied automatic differentiation to the general purpose finite element package SEPRAN consisting of approximately 400,000 lines of Fortran 77. The resulting differentiated version is produced in an automated way by augmenting the original version by additional statements computing derivatives. For a classic fluid flow experiment, we showed the improved functionality including its ease of use. Moreover, we compared the values obtained from automatic differentiation with those produced by numerical differentiation based on divided differences. The latter approach is a sensitive approximation process inherently involving the choice of a suitable step size. On the contrary, there is no concept of a step size in automatic differentiation because it accumulates derivatives of known elementary operations, finally leading to exact derivatives. For the numerical fluid flow experiment, we also showed that automatic differentiation is more efficient in terms of execution times than divided differences while only moderately increasing storage requirement.

Besides the basic features presented in this note, automatic differentiation and the software tools implementing the technology offer even more functionality. One of the highlights of automatic differentiation is the fact that a particular way to accumulate the final derivatives, the so-called reverse mode, can deliver the gradient of a scalar-valued function at a cost proportional to the function evaluation itself. That is, its cost is *independent from the number of unknowns*, whereas the cost for divided differences is roughly proportional to the gradient's length. For purposes different from mere sensitivity analysis, derivatives of arbitrary order and directional derivatives can also be obtained with similar techniques.

References

[1] M. Berz, C. Bischof, G. Corliss, and A. Griewank. *Computational Differentiation: Techniques, Applications, and Tools.* SIAM, Philadelphia, 1996.

[2] C. Bischof, A. Carle, P. Khademi, and A. Mauer. ADIFOR 2.0: Automatic differentiation of Fortran 77 programs. *IEEE Computational Science & Engineering*, 3(3):18–32, 1996.

[3] E. G. T. Bosch and C. J. M. Lasance. High accuracy thermal interface resistance measurement using a transient method. *Electronics Cooling Magazine*, 6(3), 2000.

[4] G. Corliss, A. Griewank, C. Faure, L. Hascoët, and U. Naumann, editors. *Automatic Differentiation 2000: From Simulation to Optimization*. Springer, 2001. To appear.

[5] A. Griewank. *Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation*. SIAM, Philadelphia, 2000.

[6] A. Griewank and G. Corliss. *Automatic Differentiation of Algorithms*. SIAM, Philadelphia, 1991.

[7] G. Segal. *SEPRAN Standard Problems*. Ingenieursbureau Sepra, Leidschendam, NL, 1993.

[8] G. Segal. *SEPRAN Users Manual*. Ingenieursbureau Sepra, Leidschendam, NL, 1993.

[9] G. Segal, C. Vuik, and F. Vermolen. A conserving discretization for the free boundary in a two-dimensional Stefan problem. *Journal of Computational Physics*, 141(1):1–21, 1998.

[10] A. P. van den Berg, P. E. van Keken, and D. A. Yuen. The effects of a composite non-Newtonian and Newtonian rheology on mantle convection. *Geophys. J. Int.*, 115:62–78, 1993.

[11] P. van Keken, D. A. Yuen, and L. Petzold. DASPK: a new high order and adaptive time-integration technique with applications to mantle convection with strongly temperature- and pressure-dependent rheology. *Geophysical & Astrophysical Fluid Dynamics*, 80:57–74, 1995.

[12] P. E. van Keken, C. J. Spiers, A. P. van den Berg, and E. J. Muyzert. The effective viscosity of rocksalt: implementation of steady-state creep laws in numerical models of salt diapirism. *Tectonophysics*, 225:457–476, 1993.

[13] N. J. Vlaar, P. E. van Keken, and A. P. van den Berg. Cooling of the Earth in the Archaean: consequences of pressure-release melting in a hot mantle. *Earth Plan. Sci. Lett.*, 121:1–18, 1994.

[14] C. Vuik, A. Segal, and F. J. Vermolen. A conserving discretization for a Stefan problem with an interface reaction at the free boundary. *Computing and Visualization in Science*, 3(1/2):109–114, 2000.

Parallel Factorizations with Algorithmic Blocking

Jaeyoung Choi

School of Computing, Soongsil University, Seoul, KOREA

Abstract. Matrix factorization algorithms such as LU, QR, and Cholesky, are the most widely used methods for solving dense linear systems of equations, and have been extensively studied and implemented on vector and parallel computers. In this paper, we present parallel LU, QR, and Cholesky factorization routines with an "algorithmic blocking" on 2-dimensional block cyclic data distribution. With the algorithmic blocking, it is possible to obtain the near optimal performance irrespective of the physical block size. The routines are implemented on the SGI/Cray T3E and compared with the corresponding ScaLAPACK factorization routines.

1 Introduction

In many linear algebra algorithms the distribution of work may become uneven as the algorithm proceeds, for example as in LU factorization algorithm [7], in which rows and columns are successively eliminated from the computation. The way in which a matrix is distributed over the processors has a major impact on the load balance and communication characteristics of a parallel algorithm, and hence largely determines its performance and scalability.

The two-dimensional block cyclic data distribution [9], in which matrix blocks separated by a fixed stride in the row and column directions are assigned to the same processor, has been used as a general purpose basic data distribution for parallel linear algebra software libraries because of its scalability and load balance properties. And most of the parallel version of algorithms have been implemented on the two-dimensional block cyclic data distribution [5,13].

Since parallel computers have different performance ratios of computation and communication, the optimal computational block sizes are different from one another to generate the maximum performance of an algorithm. The data matrix should be distributed with the machine specific optimal block size before the computation. Too small or large a block size makes getting good performance on a machine nearly impossible. In such case, getting a better performance may require a complete redistribution of the data matrix.

The matrix multiplication, $\mathbf{C} \Leftarrow \mathbf{C} + \mathbf{A} \cdot \mathbf{B}$, might be the most fundamental operation in linear algebra. Several parallel matrix multiplication algorithms have been proposed on the two-dimensional block-cyclic data distribution [1,6, 8,12]. High performance, scalability, and simplicity of the parallel matrix multiplication schemes using rank-K updates has been demonstrated [1,12]. It is

V.N. Alexandrov et al. (Eds.): ICCS 2001, LNCS 2073, pp. 802–811, 2001.
© Springer-Verlag Berlin Heidelberg 2001

assumed that the data matrices are distributed on the two-dimensional block cyclic data distribution and the column block size of **A** and the row block size of **B** are K. However getting a good performance when the block size is very small or very large is difficult, since the computation are not effectively overlapped with the communication. The LCM (Least Common Multiple) concept has been introduced to DIMMA [6] to use a computationally optimal block size irrespective of the physically distributed block size for the parallel matrix multiplication. In DIMMA, if the physical block size is smaller than the optimal block size, the small blocks are combined into a larger block. And if the physical block size is larger than the optimal block size, the block is divided into smaller pieces. This is the "algorithmic blocking" strategy.

There have been several efforts to develop parallel factorization algorithms with the algorithmic blocking on distributed-memory concurrent computers. Lichtenstein and Johnsson [11] developed and implemented block-cyclic order elimination algorithms for LU and QR factorization on the Connection Machine CM-200. They used a cyclic order elimination on a block data distribution, the only scheme that the Connection Machine system compilers supported.

P. Bangalore [3] has tried to develop a data distribution-independent LU factorization algorithm. He recomposed computational panels to obtain a computationally optimal block size, but followed the original matrix ordering. According to the results, the performance is superior to the other case, in which the matrix is redistributed when the block size is very small. He used a tree-type communication scheme to make computational panels from several columns of processors. However, using a pipelined communication scheme, if possible, which overlaps communication and computation effectively, would be more efficient.

The actual algorithm which is selected at runtime depending on input data and machine parameters is called "polyalgorithms" [4]. We are developing "PoLAPACK" (Poly LAPACK) factorization routines, in which computers select the optimal block size at run time according to machine characteristics and size of data matrix. In this paper, we expanded and generalized the idea in [11]. We developed and implemented parallel LU, QR, and Cholesky factorization routines with the algorithmic blocking on the 2-dimensional block cyclic data distribution. With PoLAPACK, it is always possible to have the near optimal performance of LU, QR, and Cholesky factorization routines on distributed-memory computers irrespective of the physical data-distribution on distributed-memory concurrent computers if all of the processors have the same size of submatrices.

The PoLAPACK LU, QR, and Cholesky factorization routines are implemented on the SGI/Cray T3E at KISTI Supercomputing Center, Korea. And their performance is compared with that of the corresponding ScaLAPACK factorization routines.

2 PoLAPACK LU Factorization Algorithm

The basic LU factorization routine is to find the solution vector x after applying LU factorization to A from the following linear equation, $Ax = b$. After

converting A to $P \cdot A = L \cdot U$, compute y from $L\,y = b_0$, where $U \cdot x = y$ and $P \cdot b = b_0$. And compute x.

Most of the LU factorization algorithms including LAPACK [2] and ScaLAPACK [7] find the solution vector x after computing the factorization of $P \cdot A = L \cdot U$. And in the ScaLAPACK factorization routines, a column of processors performs a factorization on its own column of blocks, and broadcasts it to others. Then all of processors update the rest of the data matrix. The basic unit of the computation is the physical size of the block, with which the data matrix is already distributed over processors.

We measured the performance the ScaLAPACK LU factorization routine and its solution routine with various block sizes on the SGI/Cray T3E. Figure 1 shows the performance on an 8×8 processor grid from $N = 1,000$ to 20,000 with block sizes of $N_b = 1, 6, 24, 36$, and 60. It shows that the near optimal performance is obtained when $N_b = 60$, and almost the same but slightly slower when $N_b = 36$ or 24. The performance deteriorated by 40% when $N_b = 6$ and 85% when $N_b = 1$. If the data matrix is distributed with $N_b = 1$, it may be much more efficient to perform the factorization after redistributing the data matrix with the optimal block size.

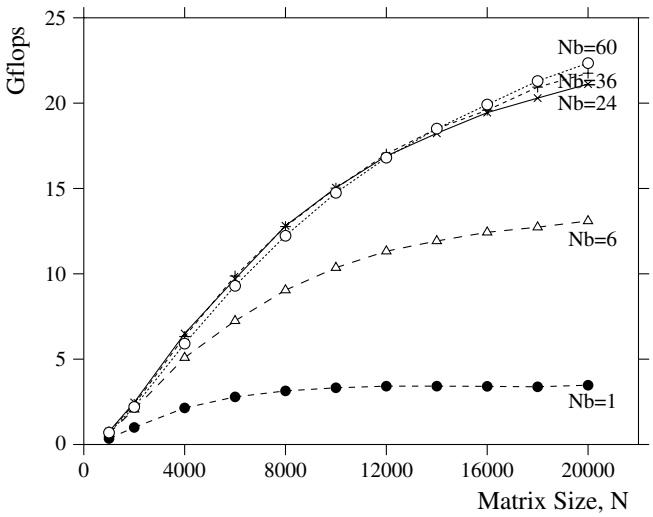


Fig. 1. Performance of ScaLAPACK LU factorization routine on an 8×8 SGI/Cray T3E

In ScaLAPACK, the performance of the algorithm is greatly affected by the block size. However the PoLAPACK LU factorization is implemented with the concept of algorithmic blocking and always shows the best performance of $N_{opt} = 60$ irrespective of physical block sizes.

If a data matrix A is decomposed over 2-dimensional $p \times q$ processors with the block cyclic data distribution, it may be possible to regard the matrix A being

decomposed along the row and column directions of processors. Then the new decomposition along the row and column directions are the same as applying permutation matrices from the left and the right, respectively. One step further. If we want to compute a matrix with a different block size, we may need to redistribute the matrix, and we can assume that the redistributed matrix is of the form $P_p \cdot A \cdot P_q^T$, where P_p and P_q are permutation matrices. It may be possible to avoid redistributing the matrix physically if the new computation doesn't follow the given ordering of the matrix A. That is, by assuming that the given matrix A is redistributed with a new optimal block size and the resulting matrix is $P_p \cdot A \cdot P_q^T$, it is now possible to apply the factorization to A with the optimal block size for the computation. And this factorization will show the same performance regardless of the physical block sizes if each processor gets the same size of the submatrix of A. These statements are illustrated with the following equations,

$$(P_p\, A\, P_q^T) \cdot (P_q x) \;=\; P_p \cdot b. \tag{1}$$

Let $A_1 = P_p\, A\, P_q^T$, and $x_1 = P_q\, x$. After factorizing $P_1 A_1 = P_1 \cdot (P_p A P_q^T) = L_1 \cdot U_1$, then we compute the solution vector x. The above equation Eq. 1 is transformed as follows:

$$L_1 \cdot U_1 \cdot (P_q x) \;=\; L_1 \cdot U_1 \cdot x_1 \;=\; P_1 \cdot (P_p b) \;=\; b_1.$$

Then, y_1 is computed from

$$L_1 \cdot y_1 \;=\; b_1, \tag{2}$$

and x_1 is computed from

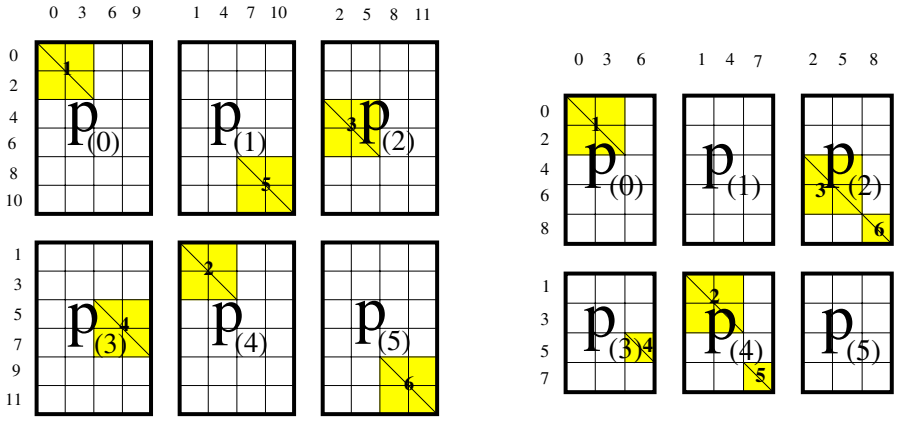
$$U_1 \cdot x_1 \;=\; y_1. \tag{3}$$

Finally the solution vector x is computed from

$$P_q \cdot x \;=\; x_1. \tag{4}$$

The computations are performed with A and b in place with the optimal block size, and x is computed with P_q as in Eq. 4. But we want $P_p \cdot x$ rather than x in order to make x have the same physical data distribution as b. That is, it is required to compute

$$P_p \cdot x \;=\; P_p \cdot P_q^T \cdot x_1. \tag{5}$$



(a) 12 x 12 blocks on 2 x 3 processes

(b) 9 x 9 blocks on 2 x 3 processes

Fig. 2. Computational Procedure in PoLAPACK. Matrices of 12×12 and 9×9 blocks are distributed on 2×3 processors with $N_{opt} = N_b$ and $N_{opt} = 2 \cdot N_b$, respectively.

3 Implementation of PoLAPACK LU Factorization

Figure 2 shows the computational procedure of the PoLAPACK LU factorization. It is assumed that a matrix A of 12×12 blocks is distributed over a 2×3 processor grid as in Figure 2(a), and the LU routine computes 2 blocks at a time (imagine $N_b = 4$ but $N_{opt} = 8$). Since the routine follows the 2-D block cyclic ordering, the positions of the diagonal blocks are regularly changed by incrementing one column and one row of processors at each step. However, if A is 9×9 blocks as in Figure 2(b), the next diagonal block of $A(5, 6)$ on $p_{(3)}$ is $A(7, 7)$ on $p_{(4)}$, not on $p_{(1)}$. Then the next block is $A(8, 8)$ on $p_{(2)}$. The computational procedure of the PoLAPACK is very complicated.

We implemented the Li and Coleman's algorithm [10] on a two dimensional processor grid for the PoLAPACK routines. But the implementation is much more complicated since the diagonal block may not be located regularly if p is not equal to q as in Figure 2.

Though p is equal to q, the implementation is still complicated. Figure 3(a) shows a snapshot of the Li and Coleman's algorithm from the processors point-of-view, where 9×9 blocks of an upper triangular matrix T are distributed over a 3×3 processor grid with $N_b = N_{opt} = 1$. Let's look over the details of the algorithm to solve $x = T \setminus b$.

At first, the last block at $p_{(8)}$ computes $x(9)$ from $T(9, 9)$ and $b(9)$. Processors in the last column update 2 blocks - actually $p_{(2)}$ and $p_{(5)}$ update $b(7)$ and $b(8)$, respectively - and send them to their left processors. The rest of b ($b(1 : 6)$) is updated later. At the second step, $p_{(4)}$ computes $x(8)$ from $T(8, 8)$ and $b(8)$, the latter is received from $p_{(5)}$. While $p_{(1)}$ receives $b(7)$ from $p_{(2)}$, updates it, and

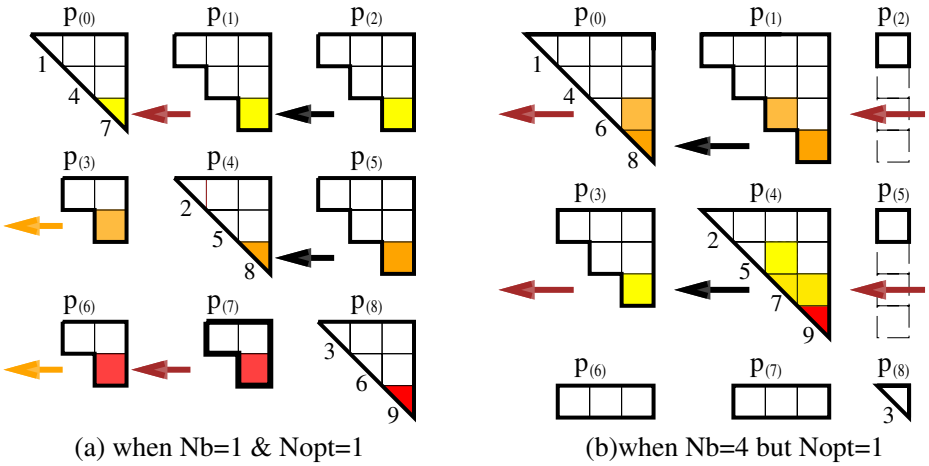


Fig. 3. A snapshot of PoLAPACK solver. A matrix T of 9×9 blocks is distributed on 3×3 processors with $N_b = 1$ and $N_b = 4$, respectively, while the optimal computational block size for both cases is $N_{opt} = 1$.

sends it to $p_{(0)}$, $p_{(7)}$ updates a temporal $b(6)$ and sends it to $p_{(6)}$.

Figure 3(b) shows the same size of the matrix distribution T with $N_b = 4$, but it is assumed that the matrix T is derived with an optimal block size $N_{opt} = 1$. So the solution routine has to solve the triangular equations of Eq. 2 and Eq. 3 with $N_{opt} = 1$. The first two rows and the first two columns of processors have 4 rows and 4 columns of T, respectively, while the last row and the last column have 1 row and 1 column, respectively. Since $N_{opt} = 1$, the computation starts from $p_{(4)}$, which computes $x(9)$. Then $p_{(1)}$ and $p_{(4)}$ update $b(8)$ and $b(7)$, respectively, and send them to their left. The rest of b ($b(1:6)$) is updated later. At the next step, $p_{(0)}$ computes $x(8)$ from $T(8,8)$ and $b(8)$, the latter is received from $p_{(1)}$. While $p_{(3)}$ receives $b(7)$ from $p_{(4)}$, updates it, and sends it to the left $p_{(5)}$, $p_{(0)}$ updates a temporal $b(6)$ and sends it to its left $p_{(2)}$. However $p_{(2)}$ and $p_{(5)}$ don't have their own data to update or compute at the current step, and hand them over to their left without touching the data. The PoLAPACK solver has to comply with this kind of all abnormal cases.

It may be necessary to redistribute the solution vector x to $P_p \cdot P_q^T \cdot x$ as in Eq. 5. However, if p is equal to q, then P_p becomes P_q, and $P_p \cdot P_q^T \cdot x = x$, therefore, the redistribution is not necessary. But if p is not equal to q, the redistribution of x is required to get the solution with the same data distribution as the right hand vector b. And if p and q are relatively prime, then the problem is changed to all-to-all personalized communication.

Figure 4 shows a case of the physical block size $N_b = 1$ and the optimal block size $N_{opt} = 2$ on a 2×3 processor grid. Originally the vector b is distributed with $N_b = 1$ as the ordering on the left of Figure 4. But the solution vector x

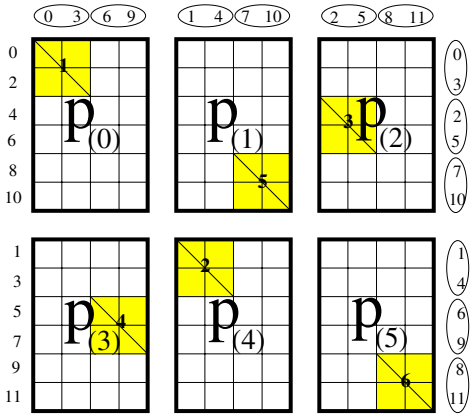


Fig. 4. A snapshot of PoLAPACK solver. A matrix T of 9×9 blocks is distributed on 3×3 processors with $N_b = 1$ and $N_b = 4$, respectively, while the optimal computational block size for both cases is $N_{opt} = 1$.

is distributed as the ordering on the right after the computation with $N_{opt} = 2$. The result is the same as a vector on the left is transposed twice – at first transposed with $N_b = 1$ to the vector on the top, then later transposed with $N_{opt} = 2$ to the vector on the right.

We implemented the PoLAPACK LU factorization routine and measured its performance on an 8×8 processor grid of the SGI/Cray T3E. Figure 5 shows the performance of the routine with the physical block sizes of $N_b = 1, 6, 24, 36,$ and 60, but the optimal block size of $N_{opt} = 60$. The performance lines are very close to the others and always show nearly the maximum performance irrespective of the physical block sizes. Since all processors don't have the same size of the submatrices of A with various block sizes, some processors have more data to compute than others, which causes computational load imbalance among processors and slight performance degradation.

4 PoLAPACK QR and Cholesky Factorization

The PoLAPACK QR factorization and its solution of the factored matrix equations are performed in a manner analogous to the PoLAPACK LU factorization and the solution of the triangular systems.

Figure 6 shows the performance of the ScaLAPACK and PoLAPACK QR factorizations and their solution on an 8×8 processor grid of the SGI/Cray T3E. Performance of the ScaLAPACK QR factorization routine depends on the physical block size, and the best performance is obtained when $N_b = 24$ on a SGI/Cray T3E. However the PoLAPACK QR factorization routine, which computes with the optimal block size of N_{opt}, always shows nearly the maximum performance independent of physical block sizes.

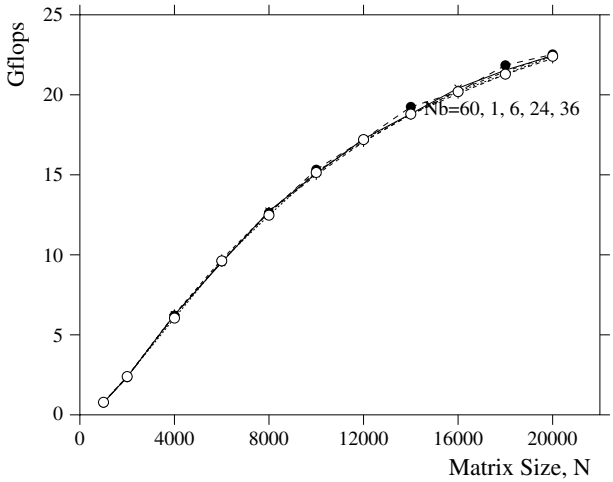


Fig. 5. Performance of PoLAPACK LU on an 8×8 SGI/Cray T3E

The Cholesky factorization factors an $N \times N$, symmetric, positive-definite matrix A into the product of a lower triangular matrix L and its transpose, i.e., $A = LL^T$ (or $A = U^T U$, where U is upper triangular).

Though A is symmetric, $P_p A P_q^T$ is not symmetric if $p \not= q$. That is, if $P_p A P_q^T$ is not symmetric, it is impossible to exploit the algorithmic blocking technique to the Cholesky factorization routine as used in the PoLAPACK LU and QR factorization. If $p \not= q$, the PoLAPACK Cholesky computes the factorization with the physical block size. That is, it computes the factorization as the same way of the ScaLAPACK Cholesky routine. However, it is possible to obtain the benefit of algorithmic blocking for the limited case of $p = q$.

Figure 7 shows the performance of the ScaLAPACK and the PoLAPACK Cholesky factorization and their solution on an 8×8 processor grid of the SGI/Cray T3E. Similarly, the performance of the ScaLAPACK Cholesky factorization routine depends on the physical block size. However the PoLAPACK Cholesky factorization routine, which computes with the optimal block size of $N_{opt} = 60$, always shows the maximum performance.

5 Conclusions

Generally in most parallel factorization algorithms, a column of processors performs the factorization on a column of blocks of A at a time, whose block size is already fixed, and then the other processors update the rest of the matrix. If the block size is very small or very large, then the processors can't show their optimal performance, and the data matrix may be redistributed for a better performance. The computation follows the original ordering of the matrix.

It may be faster and more efficient to perform the computation, if possible, by combining several columns of blocks if the block size is small, or by splitting

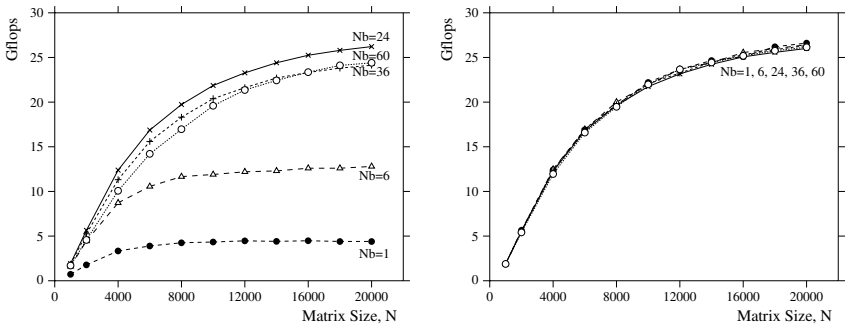


Fig. 6. Performance of ScaLAPACK QR and PoLAPACK QR on an 8×8 SGI/Cray T3E

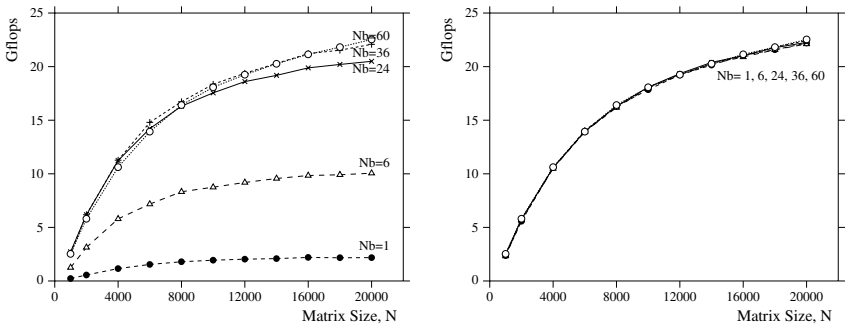


Fig. 7. Performance of ScaLAPACK and PoLAPACK Cholesky on an 8×8 SGI/Cray T3E

a large column of blocks if the block size is large. This is the main concept of algorithmic blocking. The PoLAPACK factorization routines rearrange the ordering of the computation. They compute $P_p A P_q^T$ instead of directly computing A. The computation proceeds with the optimal block size without physically redistributing A. And the solution vector x is computed by solving triangular systems, then converting x to $P_p P_q^T x$. The final rearrangement of the solution vector can be omitted if $p = q$ or $N_b = N_{opt}$.

According to the results of the ScaLAPACK and the PoLAPACK LU, QR, and Cholesky factorization routines on the SGI/Cray T3E, the ScaLAPACK factorization routines have a large performance difference with different values of N_b, however the PoLAPACK factorizations always show a steady performance, which is the near optimal, irrespective of the values of N_b. The routines we presented in this paper are developed based on the block cyclic data distribution. This simple idea can be easily applied to the other data distributions. But it is required to develop specific algorithms to rearrange the solution vector for each distribution.

References

1. R. C. Agarwal, F. G. Gustavson, and M. Zubair. A High-Performance Matrix-Multiplication Algorithm on a Distributed-Memory Parallel Computer Using Overlapped Communication. *IBM Journal of Research and Development*, 38(6):673–681, 1994.
2. E. Anderson, Z. Bai, C. Bischof, J. Demmel, J. Dongarra, J. DuCroz, A. Greenbaum, S. Hammarling, A. McKenney, and D. Sorensen. LAPACK: A Portable Linear Algebra Library for High-Performance Computers. In *Proceedings of Supercomputing '90*, pages 1–10. IEEE Press, 1990.
3. P. V. Bangalore. The Data-Distribution-Independent Approach to Scalable Parallel Libraries. 1995. Master Thesis, Mississippi State University.
4. L. Blackford, J. Choi, A. Cleary, J. Demmel, I. Dhillon, J. Dongarra, S. Hammarling, G. Henry, A. Petitet, K. Stanley, D. Walker, and R. Whaley. ScaLAPACK: A Portable Linear Algebra Library for Distributed Memory Computers - Design Issues and Performance. In *Proceedings of SIAM Conference on Parallel Processing*, 1997.
5. L. Blackford, J. Choi, E. D'Azevedo, J. Demmel, I. Dhillon, J. Dongarra, S. Hammarling, G. Henry, A. Petitet, K. Stanley, D. Walker, and R. Whaley. *ScaLAPACK Users' Guide*. SIAM Press, Philadelphia, PA, 1997.
6. J. Choi. A New Parallel Matrix Multiplication Algorithm on Distributed-Memory Concurrent Computers. *Concurrency: Practice and Experience*, 10:655–670, 1998.
7. J. Choi, J. J. Dongarra, S. Ostrouchov, A. P. Petitet, D. W. Walker, and R. C. Whaley. The Design and Implementation of the ScaLAPACK LU, QR, and Cholesky Factorization Routines. *Scientific Programming*, 5:173–184, 1996.
8. J. Choi, J. J. Dongarra, and D. W. Walker. PUMMA: Parallel Universal Matrix Multiplication Algorithms on Distributed Memory Concurrent Computers. *Concurrency: Practice and Experience*, 6:543–570, 1994.
9. V. Kumar, A. Grama, A. Gupta, and G. Karypis. *Introduction to Parallel Computing*. The Benjamin/Cummings Publishing Company, Inc., Redwood City, CA, 1994.
10. G. Li and T. F. Coleman. A Parallel Triangular Solver for a Distributed-Memory Multiprocessor. *SIAM J. of Sci. Stat. Computing*, 9:485–502, 1986.
11. W. Lichtenstein and S. L. Johnsson. Block-Cyclic Dense Linear Algebra. *SIAM J. of Sci. Stat. Computing*, 14(6):1259–1288, 1993.
12. R. van de Geijn and J. Watts. SUMMA Scalable Universal Matrix Multiplication Algorithm. LAPACK Working Note 99, Technical Report CS-95-286, University of Tennessee, 1995.
13. R. A. van de Geijn. *Using PLAPACK*. The MIT Press, Cambridge, 1997.

Bayesian Parameter Estimation: A Monte Carlo Approach

Ray Gallagher

Department of Computer Science, University of Liverpool, Liverpool L69 7ZF, United Kingdom.
Email addresses: rayg@csc.liv.ac.uk

Tony Doran

Department of Computer Science, University of Liverpool, Liverpool L69 7ZF, United Kingdom.
Email addresses: tdustdum@csc.liv.ac.uk

Abstract. This paper presents a Bayesian approach, using parallel Monte Carlo modelling algorithms for combining expert judgements when there is inherent variability amongst these judgements. The proposed model accounts for the situation when the derivative method for finding the maximum likelihood breaks down

Introduction

An expert is deemed to mean a person with specialised knowledge about a given subject area or matter of interest. This paper concerns itself with the situation where we are interested in an uncertain quantity or event and expert opinion is sort out by a decision-maker. The question then arises as to how a decision-maker should then make optimal use of the expert opinion available to them. Moreover, how does a decision-maker make optimal use of expert opinion when several experts are available to them and further resolve conflicting opinions amongst the group of experts. The opinions of an expert may come in many ways: a point estimate, parameters of uncertainty distribution or a "best guess" with upper and lower bounds. The challenge for the decision-maker is to correctly take full advantage of the data provided.

Formally uncertainty can be represented in terms of probability and the ultimate aim is to reach a consensus to arrive at a probability distribution for the uncertain quantity of interest. This distribution should fully reflect the information provided by the experts.

Various consensus procedures for the pooling of experts' opinions and probability distributions have been suggested, encompassing merely the simple averaging of

V.N. Alexandrov et al. (Eds.): ICCS 2001, LNCS 2073, pp. 812-822, 2001.
© Springer-Verlag Berlin Heidelberg 2001

expert probability distributions through to a formal Bayesian approach. Bayesian methods have been favoured by a number of researchers. Reviews of the available literature being provided by French[1], Cooke[2] together with Genest and Zidek[3]. The models proposed include those by Lindley[4-6],Morris[7,8],Winkler[9,10] and Mosley[11] This paper examines two different methods that allow the decision-maker to make the optimum decision based on available expert opinion. The methods are:

- **Derivative Method**
- **Monte Carlo**

Making the optimal decision based on the derivative method means that the function must be differentiable. We note there are other methods, discussed in Zacks,[12] to address this situation. If the function is not differentiable then we must employ a numerical method (in our case Monte Carlo) to arrive at an estimate of the quantity of interest. We further make use of parallel architectures using MIMD methods to increase the efficiency of the Monte Carlo method in situations where we may have a large body of expert opinion available.

Uncertainty Modelling of Expert Opinion

Suppose we have a parameter $\vec{\theta}$ = (θ_1, θ_2,..... θ_n) and to obtain the best decision about $\vec{\theta}$ we have to use some expert opinion given by $E = \{x_1^*, x_2^*, ..., x_N^*\}$ where x_i is the estimate of the i^{th} expert for an unknown quantity x, with the recognition that the particular value being estimated by that expert may be different from that being estimated by another expert.
The quantity of interest may be a fixed parameter but its exact value is unknown such as the height of a building or it may be an inherently variable quantity such as the IQ's of individual members of a group of people.

The situation arises, for example, when experts provide estimates based on experience with sub-populations of a non-homogeneous population. The objective is to develop an estimate of the distribution representing the variability of *x* in light of the evidence presented.

We attempt to aggregate these expert opinions to reach the "best" decision based on the estimation of $\vec{\theta}$.

For simplification we restrict ourselves to the situation when $\vec{\theta}$ comprises one or two elements. We then provide a general solution for $\vec{\theta}$ dependent on N elements. For formalisation of this discussion we consider the Bayesian approach to probability.

Let us consider the following definition of Bayes's Theorem

$$\pi(\mu \mid E) = k^{-1} L(E \mid \mu)\pi_0(\mu)$$

Where:
$\theta \equiv$ ***The value of interest*** to the decision maker,
$E \equiv$ the set of ***experts' opinions*** about the value of θ, the decision-maker treats this set of opinions as evidence/data,
$\pi_0(\theta) \equiv$ ***the decision maker's prior state*** of knowledge on θ,
$\pi(\mu|E) \equiv$ ***the decision maker's posterior state*** of knowledge on θ,
$L(E|\theta) \equiv$ ***the likelihood of observing*** the evidence E, given that the true value of the unknown quantity is θ,
$k \equiv P(E)$, the normalisation factor that makes $\pi(\theta|E)$ a probability distribution.

The problem of expert opinion is thus reduced to the assessment of the prior, π_0, and the likelihood, L, by the decision-maker. The key element in this approach is the likelihood. The likelihood function is the decision maker's tool to measure the accuracy of the expert's estimate after considering the expert's level of pertinent experience, calibration as an assessor, any known bias, and dependence to other experts.

In this section of the paper we summarise how we can receive $\pi(\hat{\theta} \mid E)$ i.e. with regard to experience, what is the best decision depends on E. Since every x_i^* is just some information concerning x_i we consider $f(x_i|\theta)$ as the actual distribution of the quantity of interest, x. We consider $L(x_i^* \mid \theta)$ is the probability density that the experts' estimate is x_i^* if the decision maker believes that the i^{th} expert is perfect then $L(x_i^* \mid \theta) = f(x_i|\theta)$.

Since the experts are considered independent then we have

$$L(E \mid \theta) = L(x_1^*, x_2^*, ..., x_n^* \mid \theta) = \prod_{i=1}^{n} L(x_i^* \mid \theta) \tag{1}$$

Moreover, $\pi(\theta \mid x_1^*, x_2^*, ..., x_N^*) = k^{-1} L(x_1^*, x_2^*, ...x_n^* \mid \theta)\pi_0(\theta)$. In this method we should first obtain k such that $\pi(\theta \mid x_1^*, x_2^*, ...x_n^*)$ is the conditional distribution. Suppose $P_i = P_i(x_i^* \mid x_i)$, (this P_i is one, if and only if, the expert is considered to be perfect) is the probability that the i^{th} expert says x_i^* when in fact the true value is x_i. The quantity P_i is the decision maker's probability density that the expert's estimate is x_i^* when he is attempting to estimate x_i.

We should note that x_i is one possible value of x and x is distributes according to $f(x|\theta)$. Then

$$L_i(x_i^* \mid \theta) = \begin{cases} \int P_i(x_i^* | xf(x|\theta)dx & if \quad X \quad continuous \\ \sum_j P(x_i^* | x_j) P(x_j | P) & if \quad X \quad discrete \end{cases} \tag{2}$$

For N independent experts we have

$$\pi(\theta \mid x_1^*, x_2^*, \ldots . x_n^*) = \begin{cases} k^{-1}\left\{\int P_i(x_i^* | x) f(x|\theta)dx\right\}\pi_0(\theta) \\ k^{-1}\left\{\prod_{i=1}^{n}\sum_j P_i(x_i^* | x_j) f(x_j|\theta)\right\}\pi_0(\theta) \end{cases} \tag{3}$$

For the best decision based on the evidence, *E,* we can use the derivative method if the derivative exists i.e.

$$\frac{\partial}{\partial \theta_j}\pi(\theta \mid x_1^*, x_2^*, \ldots . x_n^*) = 0 \quad j = 1,2,\ldots,n \tag{4}$$

These systems named normal equations, and receive $\theta_j = \hat{\theta}_j$ and for the maximum of *L* must be

$$\frac{\partial^2}{\partial \theta_j^2}\pi(\theta \mid x_1^*, x_2^*, \ldots . x_n^*)\Big|_{\theta_j = \hat{\theta}_j} < 0 \quad j = 1,2,\ldots,n \tag{5}$$

Example:

Suppose the decision-maker is interested in assessing the probability distribution of a random variable that takes only two values i.e. let

$$X = \{x_1, x_2\}. \tag{6}$$

A discrete distribution of X is completely known if we know P, where
$\theta \equiv \Pr[X = x_1]$ and $1 - \theta \equiv \Pr[X = x_2]$, $0 \le \theta \le 1$.
Suppose now the decision-maker asks the opinion of *N* experts on whether $X = x_1$ or whether $X = x_2$. Let *E*, defined as $E = \{x_1^*, x_2^*, ..., x_N^*\}$ be the set of expert responses where x_i^*, the i^{th} response can be either $X = x_1$ or $X = x_2$. Then we have $\pi(\theta \mid E) = k^{-1} L(E \mid \theta)\pi_0(\theta)$ where

$$L(E \mid \theta) = \prod_{i=1}^{n} L_i(x_i^* \mid \theta) \text{ and } L(x_i^* \mid \theta) = \sum_{j=1}^{n} P_r(x_i^* \mid x_j) P_r(x_j \mid \theta). \quad (7)$$

It is trivial that

$$P_r(x_j \mid \theta) = \begin{cases} \theta & if \quad j = 1 \\ 1 - \theta & if \quad j = 2 \end{cases} \quad (8)$$

Where $\Pr(x_i^* \mid x_j)$, is the probability that the ith expert says x_i^* when in fact $X = x_j$. These values represent how good the decision-maker thinks the experts are. For example, let us assume that the decision-maker consults two experts who he believes to be perfect and independent. For simplicity we assume a uniform prior in the closed interval [0,1], i.e. $\pi_0(\theta) = 1$, and consider the following two cases.

Case (i)

The two experts have opposing opinions, e.g. $x_1^* = x_1$ and $x_2^* = x_2$. Then, the likelihood is

$$L = \prod_{i=1}^{n} L_i(x_i \mid \theta) = \theta(1 - \theta) \quad (9)$$

and the posterior will be:

$$\pi(\theta \mid x_1, x_2) = 6\theta(1 - \theta) \quad (10)$$

With regard to equation (9) we have $\pi(\theta \mid E) = k^{-1}\theta(1-\theta)$ since π(θ|E) should be a conditional distribution then

$$\int_0^1 \pi(\theta \mid E)d\theta = k^{-1} \Rightarrow k = 1/6 \tag{11}$$

Then we have

$$\pi(\theta \mid x_1^*, x_2^*, ...x_n^*) = 6\theta(1-\theta) = 6\theta - 6\theta^2 \qquad 0 \le \theta \le 1 \tag{12}$$

Now, with regards to derivative tests for finding the extreme points we have

$$\frac{\partial \pi}{\partial \theta} = 6 - 12\theta = 0 \Rightarrow \hat{\theta} = 1/2 \tag{13}$$

$$\left.\frac{\partial^2 \pi}{\partial \theta^2}\right|_{\hat{\theta}=\frac{1}{2}} < 0 \tag{14}$$

This represents the distribution of all possible distributions of X. The most probable distribution (i.e. the mode of the posterior $\pi(\theta|x_1,x_2)$) is given by θ=1/2. It means that starting from complete lack of knowledge about the distribution of X, the opposing opinions of two independent experts have caused the decision maker to think most probably X= x_1 and X= x_2 are equally likely.

Case (ii)

The two experts have the same opinion; that is, for example, $x_1^* = x_1$ and $x_2^* = x_1$. The posterior in this case will be

$$\pi(\theta \mid x_1, x_2) = 3\theta^2 \tag{15}$$

We leave the proof of the second case as it is essentially the same operation of case i.

The main idea of this paper is when the situation arises when we wish to arrive at the optimum decision when there is no derivative, Zacks,[12]. In this situation we can use the finite difference gradient algorithm.

$$\theta_{i+1} = \theta_i + \alpha_i \hat{\nabla} \pi(\theta \mid E) \tag{16}$$

Therefore

$$\hat{\nabla} \pi = \left(\frac{\partial \hat{\pi}}{\partial \theta_1}, \frac{\partial \hat{\pi}}{\partial \theta_2}, \ldots\ldots\ldots, \frac{\partial \hat{\pi}}{\partial \theta_n} \right) \tag{17}$$

where

$$\frac{\partial \hat{\pi}}{\partial \theta_1} = \frac{g((\theta_1 + \Delta\theta), \theta_2, \ldots\ldots, \theta_n) - g((\theta_1 - \Delta\theta), \theta_2, \ldots\ldots, \theta_n)}{2\Delta\theta} \tag{18}$$

$$i = 1,2,\ldots,n$$

In this case we can consider a Monte Carlo random search algorithm to estimate the optimum decision for θ.

Random Search

We choose the random search double trial algorithm, Rubinstein[14].

$$\theta_{i+1} = \theta_i + \frac{\alpha_i}{2\Delta\theta_i} \left[\pi(\theta_i + \Delta\theta_i t_i) - \pi(\theta_i - \Delta\theta_i t_i) \right] t_i \tag{19}$$

where α_i and $\Delta\theta_1$ are greater than 0. This estimation $\hat{\theta}$ of θ, converges to θ in quadratic mean, in probability, and with probability one, Halton[13].
This algorithm may be performed by generating the random vector t_i continuously distributed on the *n*-dimensional unit sphere.

For this algorithm, if π is a real function which depends on $\vec{\theta} = (\theta_1, \theta_2, ..., \theta_m)$, then we can use $\vec{t_i} = (t_{i1}, t_{i2}, ...t_{in})$, i = 1,...,m, and use *n* random vectors, in this situation we have a lot of random samples and we can try by parallel processing methods in a MIMD environment to obtain θ_i by the i^{th} processor, in a small time interval.

If $\vec{\theta} = (\theta_1, \theta_2, ..., \theta_m)$, we generate (i = 1,...,m) $\vec{t_i} = (t_{i1}, t_{i2}, \ldots, t_{in})$ m random vectors and

$\overrightarrow{t_{01}}$	t_{11}	t_{12}	t_{13}	...	...	...	t_{1n}
$\overrightarrow{t_{02}}$	t_{21}	t_{22}	t_{23}	...	...	...	t_{2n}
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
$\overrightarrow{t_{0m}}$	t_{m1}	t_{m2}	t_{m3}	...	...	...	t_{mn}

$\vec{t} = \begin{bmatrix} t_{01} \\ t_{02} \\ t_{03} \\ \vdots \\ t_{0n} \end{bmatrix}$ is a vector with m rows and each element is n-tuples of random

numbers. Then we collect the following random vectors

$$\vec{t}_{01} = \begin{bmatrix} t_{11} \\ t_{21} \\ t_{31} \\ \vdots \\ t_{n1} \end{bmatrix}, \quad \vec{t}_{02} = \begin{bmatrix} t_{12} \\ t_{22} \\ t_{32} \\ \vdots \\ t_{n2} \end{bmatrix} \quad ,..., \quad \vec{t}_{0m} = \begin{bmatrix} t_{1m} \\ t_{2m} \\ t_{3m} \\ \vdots \\ t_{nm} \end{bmatrix},$$

for distribution to processors *1, 2, ..., m* respectively, enabling us to obtain the result $\hat{\theta}_i$ from the i^{th} processor. We will then have $\hat{\theta} = (\hat{\theta}_1, ..., \hat{\theta}_n)$ such that $\hat{\theta}$ is an estimate for θ.

Algorithm 1

1. **Set** Sum = 0

2. **Do** N times

3. **Generate** $\vec{t_i} = (t_{i1}, t_{i2}, \ldots, t_{im})$ sampling from $\vec{z_{0i}}$

4. **Do** until convergence

5. **Set**

$$\theta_{i+1} = \theta_i + \frac{\alpha_i}{2\Delta\theta_i}\left[\pi(\theta_i + \Delta\theta_i t_i) - \pi(\theta_i - \Delta\theta_i t_i)\right] t_i$$

6. **Set** Sum = Sum + θ_{i+1}

7. **Goto 3.**

8. **Goto 2.**

9. **Set** Sum = Sum/N

10. **Set** $\hat{\theta} = \theta_n$.

Algorithm 2

1. **Get** the parameter $\vec{\theta} = (\theta_1, \ldots, \theta_m)$, and $\pi(\theta \mid E)$.

2. **Generate** the random vector $\vec{z} = (z_{11}, z_{12}, \ldots, z_{1n})$ for i = 1,2,...,m.

3. **Collect** $\vec{z}_{01} = \begin{bmatrix} z_{11} \\ z_{21} \\ z_{31} \\ \vdots \\ z_{n1} \end{bmatrix}$, $\vec{z}_{02} = \begin{bmatrix} z_{12} \\ z_{22} \\ z_{32} \\ \vdots \\ z_{n2} \end{bmatrix}$, ..., $\vec{z}_{0m} = \begin{bmatrix} z_{1m} \\ z_{2m} \\ z_{3m} \\ \vdots \\ z_{nm} \end{bmatrix}$,

4. **Send** $\vec{z}_{01}$, $\vec{z}_{02}$, ..., $\vec{z}_{0m}$ to processors 1, 2, ...,m.

5. **Algorithm 1.**

6. **Get** $\hat{\theta}_1, \hat{\theta}_2, \ldots\ldots, \hat{\theta}_m$ i.e. the Monte Carlo estimates of θ_i.

7. **Consider** $\hat{\theta} = (\hat{\theta}_1, \hat{\theta}_2, ..., \hat{\theta}_m)$, as an estimation of θ such as $\pi(\hat{\theta} \mid E)$ is an optimal estimate of $\pi(\theta \mid E)$ without the need to resort to the derivative method.

Conclusion

This approach allows a solution to be obtained when there is no derivative. Further by virtue of parallel processing it allows complex models containing many experts to be calibrated. It is intended to consider the problems of dependencies amongst experts in a further paper where the computational demands are considered to be excessive.

References

1. French, S., Group Consensus Probability Distribution: A Critical Survey. In *Bayesian Statistics* **2**, ed J. M. Bernardo, M. H. DeGroot, D. V. Lindley & A. F. M. Smith. North Holland, Amsterdam, (1985), pp. 183-201.
2. Cooke, R. M., Expert Opinion and Subjective Probability in Science. Delft University of Technology Report, Chapter 11, (1990).
3. Genest, C. & Zidek, J. V., Combining Probability Distributions: A Critique and an Annotated Bibliography. *Statistical Science*., **1** (1986) 114-48.
4. Lindley, D. V., Reconciliation of Probability Distributions. *Operational Research.*, **31** (1983) 866-80.
5. Lindley, D. V. & Singpurwalla, N., Reliability (and Fault Tree) Analysis Using Expert Opinions. *Journal American Statistical Association*., **81** (393) (1986) 87-90.
6. Lindley, D. V., Tversky, A. & Brown, R. V., On the Reconciliation of Probability Assessments (with discussion). *J. R. Statist. Soc. Ser A*, **142** (1979) 146-80.
7. Morris, P. A. Combining Expert Judgements: A Bayesian Approach. *Management Science*., **23** (1977) 679-93.
8. Morris, P. A., An Axiomatic Approach to Expert Resolution. *Management Science*., **29** (1983) 866-80.
9. Winkler, R. L., The Consensus of Subjective Probability Distributions *Management Science*., **15** (1968) B61-B75).
10. Winkler, R. L., Combining Probability Distributions from Dependent Information *Sources. Management Science*., **27** (1981) 479-88.
11. Mosley, A. Bayesian modeling of expert-to-expert variability and dependence in estimating rare event frequencies. *Reliability Engineering and System Safety* **38** (1992) 47-57.
12. Zacks, S. The Theory of Statistical Inference. John Wiley and Sons. New York (1971)
pages (230-233).
13. Halton, J. H., A retrospective survey of the Monte Carlo method. *Siam Rev*, **12(1)** (1970) 1-63.

14. Rubinstein, R.Y., Simulation and the Monte Carlo Method. John Wiley and Sons. New York (1981) Page 238.

Contact Information: Ray Gallagher. Telephone 44-151-794-3161. Facsimile 44-151-3715

Recent Progress in General Sparse Direct Solvers

Anshul Gupta

IBM T.J. Watson Research Center
P.O. Box 218, Yorktown Heights, NY 10598
anshul@watson.ibm.com
http://www.cs.umn.edu/~agupta/wsmp.html

Abstract. During the past few years, algorithmic improvements alone have shaved almost an order of magnitude off the time required for the direct solution of general sparse systems of linear equations. Combined with a similar increase in the performance to cost ratio due to hardware advances during this period, current sparse solver technology makes it possible to solve those problems quickly and easily that might have been considered impractically large until recently. In this paper, we compare the performance of some commonly used software packages for solving general sparse systems. In particular, we demonstrate the consistently high level of performance achieved by WSMP—the most recent of such solvers. We compare the various algorithmic components of these solvers and show that the choices made in WSMP enable it to run two to three times faster than the best amongst other similar solvers. As a result, WSMP can factor some of the largest sparse matrices available from real applications in a few seconds on 4-CPU workstation.

1 Introduction

Developing an efficient parallel, or even serial, direct solver for general sparse systems of linear equations is a challenging task that has been the subject of research for the past four decades. Several breakthroughs have been made during this time. As a result, a number of very competent software packages for solving such systems are available [2,4,6,8,17,23,26,25].

In this paper, we compare the performance of some commonly used software packages for solving general sparse systems and show that during the past few years, algorithmic improvements alone have shaved an order of magnitude off the time required to factor general sparse matrices. Combined with a similar increase in the performance to cost ratio due to hardware advances during this period, current sparse solver technology makes it possible to solve those problems quickly and easily that might have been considered impractically large until recently. We demonstrate the consistently high level of performance achieved by the Watson Sparse Matrix Package (WSMP) and show that it can factor some of the largest sparse matrices available from real applications in a few seconds on 4-CPU workstation. The WSMP project's original aim was to develop a scalable

V.N. Alexandrov et al. (Eds.): ICCS 2001, LNCS 2073, pp. 823–831, 2001.
© Springer-Verlag Berlin Heidelberg 2001

parallel general sparse solver for a distributed-memory parallel computer like the IBM SP. However, after completing the serial version of the solver, we realized that we couldn't find enough large problems that would justify the use of an SP with several nodes. Therefore, we tailored the parallel version to use a few CPU's in a shared-memory environment. It is one of the objectives of this paper to make the user community aware of the robustness and speed of the current sparse direct solver technology and encourage scientists and engineers to develop bigger models with larger sparse systems so that the full potential of these solvers can be utilized.

2 Comparison of Serial Performance of Some General Sparse Solvers

In this section, we compare the performance of some of the most well-known software packages for solving sparse systems of linear equations on a single CPU of an IBM RS6000 S80. This is a 600 Mhz processor with a 64 KB level-1 cache and a peak theoretical speed of of 1.2 Gigaflops. 2 GB of memory was available to each solver. A more detailed comparison of the serial and the parallel versions of these solvers can be found in [19]. Table 1 lists all the test matrices, their dimensions, number of nonzeros, and application areas of their origin.

Table 2 lists the serial LU factorization time taken by UMFPACK [8], SuperLU [9], SPOOLES [4], SuperLU$_{dist}$ [22,23], MUMPS [1,2], and WSMP [15,17]. This table also lists the year in which the latest version of each of these packages became available. The best factorization time for each matrix using any solver released before year 2000 is shown in italics and the overall best factorization time is shown in boldface.

The most striking observation in Table 2 is the range of times that different packages available before 1999 would take to factor the same matrix. It is not uncommon to notice the fastest solver being faster than the slowest one by one to two orders of magnitude. Additionally, none of them gave a consistent level of performance. For example, UMFPACK is 13 times faster than SPOOLES on *e40r5000* but 14 times slower on *fidap011*. MUMPS is clearly the fastest and the most robust amongst the solvers available before 2000. However, the latest solver WSMP appears to be about two and half times faster than MUMPS on this machine on an average. WSMP also has the most consistent performance. It has the smallest factorization time for all but two matrices and is the only solver that does not fail on any of the test matrices.

3 Algorithmic Features of the Solvers

In this section, we list the algorithms and strategies that these solvers use for the symbolic and numerical phases of the computation of the LU factors of a general sparse matrix. We then briefly discuss the role of these choices on the performance of the solvers. The description of all these algorithms is beyond

Table 1. Test matrices with their order (N), number of nonzeros (NNZ), and the application area of origin.

Matrix	*N*	*NNZ*	Application
af23560	23560	484256	Fluid dynamics
av41092	41092	1683902	F.E.M.
bayer01	57735	277774	Chemistry
bbmat	38744	1771722	Fluid dynamics
comp2c	16783	578665	Linear programming
e40r0000	17281	553956	Fluid dynamics
e40r5000	17281	553956	Fluid dynamics
ecl32	51993	380415	Electrical eng.
epb3	84617	463625	Thermodynamics
fidap011	16614	1091362	F.E.M.
fidapm11	22294	623554	F.E.M.
invextr1	30412	1793881	Fluid dynamics
lhr34c	35152	764014	Chemical eng.
lhr71c	70304	1528092	Chemical eng.
mil053	530238	3715330	F.E.M.
mixtank	29957	1995041	Fluid dynamics
nasasrb	54870	2677324	F.E.M.
onetone1	36057	341088	Circuit simulation
onetone2	36057	227628	Circuit simulation
pre2	659033	5959282	Circuit simulation
raefsky3	21200	1488768	Fluid dynamics
raefsky4	19779	1316789	Fluid dynamics
rma10	46835	2374001	Fluid dynamics
tib	18510	145149	Circuit simulation
twotone	120750	1224224	Circuit simulation
venkat50	62424	1717792	Fluid dynamics
wang3	26064	177168	Circuit simulation
wang4	26068	177196	Circuit simulation

the scope of this paper. The reader should be able to find them in the citations provided.

1. UMFPACK [8]
   - *Fill reducing ordering:* Approximate minimum degree [7] on unsymmetric structure, combined with suitable numerical pivot search during LU factorization.
   - *Task dependency graph:* Directed acyclic graph.
   - *Numerical factorization:* Unsymmetric-pattern multifrontal.
   - *Pivoting strategy:* Threshold pivoting implemented by row-exchanges.
2. SuperLU [9]
   - *Fill reducing ordering:* Multiple minimum degree (MMD) [13] on the symmetric structure of AA^T or $A+A^T$, where A is the original coefficient matrix.

Table 2. LU Factorization time on a single CPU (in seconds) for UMFPACK, SuperLU, SPOOLES, SuperLU$_{dist}$, MUMPS, and WSMP, respectively. The best pre-2000 time is shown in italics and the overall best time is shown in boldface.

Year →	1994	1997	1998	1999	1999	2000
Matrix	UMFPACK	SuperLU	SPOOLES	SuperLU$_{dist}$	MUMPS	WSMP
af23560	45.5	31.9	10.5	14.7	*8.93*	**6.19**
av41092	186.	772.	Fail	Fail	*30.6*	**8.47**
bayer01	*1.76*	2.40	Fail	3.23	2.26	**1.33**
bbmat	682.	214.	*97.7*	Fail	113.	**36.7**
comp2c	120.	3403	287.	42.0	*29.3*	**4.08**
e40r5000	29.7	43.9	395.	2.08	**1.18**	1.55
ecl32	Fail	Fail	562.	Fail	*145.*	**41.2**
epb3	29.7	24.2	*5.00*	5.67	5.69	**2.16**
fidap011	168.	39.9	*12.2*	16.9	18.7	**6.38**
fidapm11	944.	88.9	*15.1*	Fail	25.3	**11.2**
lhr71c	*6.80*	12.5	Fail	23.0	11.7	**3.05**
nasasrb	81.8	102.	*25.0*	Fail	26.8	**10.9**
onetone1	12.2	184.	113.	10.7	*10.0*	**7.25**
onetone2	*1.79*	28.3	20.7	3.55	2.81	**1.11**
pre2	Fail	Fail	Fail	Fail	Fail	**362.**
raefsky3	39.0	146.	10.0	*6.86*	8.75	**4.54**
raefsky4	109.	1983	157.	28.1	*27.5*	**7.78**
rma10	15.7	Fail	10.7	*5.78*	9.62	**3.76**
tib	*0.52*	266.	1.75	1.47	0.62	**0.31**
twotone	**30.0**	Fail	724.	637.	124.	37.9
venkat50	16.2	Fail	11.6	*8.11*	19.4	**4.40**
wang3	106.	3226	62.7	36.9	*32.3*	**13.4**
wang4	97.3	318.	*16.2*	23.7	25.6	**12.0**

- *Task dependency graph:* Tree.
- *Numerical factorization:* Supernodal Crout.
- *Pivoting strategy:* Threshold pivoting implemented by row-exchanges.

3. SPOOLES [4]
   - *Fill reducing ordering:* Generalized nested dissection/multisection [5] on the symmetric structure of $A + A^T$.
   - *Task dependency graph:* Tree.
   - *Numerical factorization:* Supernodal Crout.
   - *Pivoting strategy:* Threshold rook pivoting that may perform both row and column exchanges to control growth in both L and U.
4. SuperLU$_{dist}$ [22,23]
   - *Fill reducing ordering:* Multiple minimum degree [13] on the symmetric structure of $A + A^T$.
   - *Task dependency graph:* Directed acyclic graph.
   - *Numerical factorization:* Supernodal right-looking.

- *Pivoting strategy:* No numerical pivoting during factorization. Rows are preordered to maximize the magnitude of the product of the diagonal entries [11].

5. MUMPS [1,2]
   - *Fill reducing ordering:* Approximate minimum degree [7] on the symmetric structure of $A + A^T$.
   - *Task dependency graph:* Tree.
   - *Numerical factorization:* Symmetric-pattern multifrontal.
   - *Pivoting strategy:* Preordering rows to maximize the magnitude of the product of the diagonal entries [11], followed by unsymmetric row exchanges within supernodes and symmetric row and column exchanges between supernodes.
6. WSMP [15,17]
   - *Fill reducing ordering:* Nested dissection [18,16] on the symmetric structure of $A + A^T$.
   - *Task dependency graph:* Minimal directed acyclic graph [15].
   - *Numerical factorization:* Unsymmetric-pattern multifrontal.
   - *Pivoting strategy:* Preordering rows to maximize the magnitude of the product of the diagonal entries [11], followed by unsymmetric partial pivoting within supernodes and symmetric pivoting between supernodes. Rook pivoting (which attempts to contain growth in both L and U) is an option.

The multifrontal method [12,24] for the solving sparse systems of linear equations offers a significant performance advantage over more conventional factorization schemes by permitting efficient utilization of parallelism and memory hierarchy. Our detailed experiments in [19] show that all three multifrontal solvers—UMFPACK, MUMPS, and WSMP—run at a much higher Megaflop rate than their non-multifrontal counterparts. The original multifrontal algorithm proposed by Duff and Reid [12] uses the symmetric pattern of $A + A^T$ to generate an elimination tree to guide the numerical factorization, which works on symmetric frontal matrices. This symmetric-pattern multifrontal algorithm used in MUMPS can incur a substantial overhead for very unsymmetric matrices due to unnecessary dependencies in the elimination tree and extra zeros in the artificially symmetric frontal matrices. Davis and Duff [8] and Hadfield [21] introduced an unsymmetric-pattern multifrontal algorithm, which is used in UMFPACK and overcomes the shortcomings of the symmetric-pattern multifrontal algorithm. However, UMFPACK did not reveal the full potential of the unsymmetric-pattern multifrontal algorithm because of the choice of a fill-reducing ordering (AMD), which has now been shown to be less effective than nested dissection [3]. Moreover, the merging of the ordering and symbolic factorization within numerical factorization slowed down the latter and excluded the possibility of using a better ordering while retaining the factorization code. Other than WSMP, SPOOLES is the only solver that uses a graph-partioning based ordering. However, it appears that the fill-in resulting from rook pivoting,

which involves both row and column exchanges in an attempt to limit pivot growth in both L and U, overshadows a good initial ordering. Simple threshold partial pivoting yields a sufficiently accurate factorization for most matrices, including all our test cases. Therefore, rook pivoting is an option in WSMP, but the default is a simple threshold pivoting.

WSMP achieves superior levels of performance by incorporating the best ideas of the previous solvers into one package, while avoiding their pitfalls and by introducing new techniques in both symbolic and numerical phases. The analysis or preprocessing phase of WSMP uses a reduction to a block-triangular form [10], a permutation of rows to maximize the magnitude of the product of the diagonal entries [11,20], a multilevel nested dissection ordering [16] for fill-in reduction, and an improved version [15] of the classical unsymmetric symbolic factorization algorithm [14] to determine the supernodal structure of factors and the minimal directed acyclic task- and data-dependency graphs to guide potentially multiple steps of numerical factorization with minimum overhead and maximum parallelism. The unsymmetric-pattern multifrontal LU factorization algorithm of WSMP [15] uses novel data-structures to efficiently handle any amount of pivoting and different pivot sequences without repeating the symbolic phase for each factorization.

4 Performance on a Shared-Memory Parallel Workstation

Having empirically established in Section 2 that MUMPS and WSMP are the fastest and the most robust among the currently available general sparse solvers, in Table 3, we give 1- and 4-CPU factorization times of these two solvers on an IBM RS6000 workstation with 375 Mhz Power 3 processors. These processors have a peak theoretical speed of 1.5 Gigaflops. They share a 4 MB level-2 cache and have a 64 KB level-1 cache each. 2 GB of memory was available to each single CPU run and the 4-CPU runs of WSMP. MUMPS, when run on 4 processors, had a total of 4 GB of memory available to it.

The relative performance of WSMP improves on the Power 3 as it is able to extract a higher Megaflop rate from this machine. The most noteworthy observation from Table 3 (last column) is that out of the 25 test cases, only 3 require more than 10 seconds on a mere workstation and all but one of the matrices can be factored in under 13 seconds. The factorization times reported in Table 3 use the default options of WSMP. Many of the large test matrices, such as *fidapm11, mil053, mixtank, nasasrb, raefsky3, raefsky4, rma10, venkat50, wang3,* and *wang4* have a symmetric structure and would need even less factorization time if the user switches off the optional pre-permutation of rows to maximize the diagonal magnitudes. This row permutation, which is on by default in WSMP, destroys the structural symmetry and increases the fill-in and operation count of factorization. Some of the matrices, such as *mil053, venkat50, wang3, wang4*, etc. do not require partial pivoting to yield an accurate factorization. Therefore, if the user familiar with the characteristics of the matrices, switches off the piv-

Table 3. Number of factor nonzeros, operation count, and LU factorization (with partial pivoting) times of MUMPS and WSMP on one and four 375 Mhz Power 3 processors.

Matrix	MUMPS				WSMP			
	NNZ_{LU} $\times 10^6$	Ops $\times 10^9$	1 CPU	4 CPUs	NNZ_{LU} $\times 10^6$	Ops $\times 10^9$	1 CPU	4 CPUs
af23560	8.34	2.56	**7.75**	**3.82**	11.1	3.27	**4.17**	**2.13**
av41092	14.1	8.42	**19.8**	**10.7**	9.28	2.02	**4.71**	**2.90**
bayer01	2.82	.125	**2.23**	**1.26**	1.75	.040	**1.34**	**1.33**
bbmat	46.0	41.4	**76.3**	**32.2**	36.6	20.1	**23.7**	**8.68**
comp2c	7.05	4.22	**22.3**	**13.6**	3.53	1.09	**2.72**	**1.02**
e40r0000	1.72	.172	**1.70**	**1.20**	2.32	.250	**0.62**	**0.37**
ecl32	42.9	64.6	**82.6**	**41.0**	30.3	21.0	**23.9**	**8.71**
epb3	6.90	1.17	**5.10**	**2.49**	5.64	.451	**1.94**	**1.90**
fidap011	12.5	7.01	**14.5**	**11.8**	10.2	3.20	**4.22**	**2.01**
fidapm11	14.0	9.67	**18.2**	**10.2**	14.9	5.21	**6.77**	**2.73**
invextr1	30.3	35.6	**53.6**	**29.8**	16.3	6.90	**10.8**	**6.36**
lhr34c	5.58	.641	**5.30**	**2.43**	3.47	.170	**1.16**	**1.13**
mil053	75.9	31.8	**68.8**	**26.0**	66.2	14.4	**24.8**	**12.9**
mixtank	38.5	64.4	**80.0**	**38.4**	27.0	19.5	**22.9**	**9.34**
nasasrb	24.2	9.45	**23.0**	**16.4**	21.7	5.41	**7.22**	**3.86**
onetone1	4.72	2.29	**7.06**	**4.21**	4.34	1.79	**3.88**	**1.96**
onetone2	2.26	.510	**2.33**	**1.41**	1.62	.206	**0.90**	**0.94**
pre2	358.	Fail	**Fail**	**Fail**	97.5	133.	**189.**	**77.0**
raefsky3	8.44	2.90	**8.37**	**5.58**	9.42	2.57	**3.27**	**1.54**
raefsky4	15.7	10.9	**19.9**	**13.2**	11.8	4.11	**5.09**	**2.54**
rma10	8.87	1.40	**8.41**	**5.38**	10.9	1.48	**2.59**	**1.02**
twotone	22.1	29.3	**68.7**	**32.1**	12.5	10.4	**18.3**	**12.9**
venkat50	12.0	2.31	**9.75**	**5.27**	12.9	1.75	**3.01**	**1.24**
wang3	13.8	13.8	**20.1**	**8.34**	11.9	5.91	**6.92**	**3.31**
wang4	11.6	10.5	**16.6**	**7.67**	12.2	6.09	**7.06**	**3.51**

oting for these matrices and, in general, tailors the various options [17] to her application, many of the test problems can be solved even faster.

5 Concluding Remarks

In this paper, we show that recent sparse solvers have improved the state of the art of the direct solution of general sparse systems by almost an order of magnitude. Coupled with the good scalability of these solvers [3] and the availability of relatively inexpensive high-performance parallel computers, it is now possible to solve very large sparse systems in only a small fraction of the time that these solutions would have required just a few years ago. Judging by the availability of real test cases, it appears that the applications that require the solution of such

systems have not kept pace with the improvements in the software and hardware available to solve these systems. We hope that the new sparse solvers will encourage scientists and engineers to develop bigger models with larger sparse systems so that the full potential of the new generation of parallel sparse general solvers can be exploited.

References

1. Patrick R. Amestoy, Iain S. Duff, Jacko Koster, and J. Y. L'Execellent. A fully asynchronous multifrontal solver using distributed dynamic scheduling. Technical Report RT/APO/99/2, ENSEEIHT-IRIT (Toulouse, France), 1999. To appear in SIAM Journal on Matrix Analysis and Applications.
2. Patrick R. Amestoy, Iain S. Duff, and J. Y. L'Execellent. Multifrontal parallel distributed symmetric and unsymmetric solvers. *Computational Methods in Applied Mechanical Engineering*, 184:501–520, 2000. Also available at http://www.enseeiht.fr/apo/MUMPS/.
3. Patrick R. Amestoy, Iain S. Duff, J. Y. L'Execellent, and Xiaoye S. Li. Analysis, tuning, and comparison of two general sparse solvers for distributed memory computers. Technical Report RT/APO/00/2, ENSEEIHT-IRIT, Toulouse, France, 2000. Also available as Technical Report 45992 from Lawrence Berkeley National Laboratory.
4. Cleve Ashcraft and Roger G. Grimes. SPOOLES: An object-oriented sparse matrix library. In *Proceedings of the Ninth SIAM Conference on Parallel Processing for Scientific Computing*, March 1999.
5. Cleve Ashcraft and Joseph W.-H. Liu. Robust ordering of sparse matrices using multisection. Technical Report CS 96-01, Department of Computer Science, York University, Ontario, Canada, 1996.
6. Michel Cosnard and Laura Grigori. Using postordering and static symbolic factorization for parallel sparse LU. In *Proceedings of the International Parallel and Distributed Processing Symposium (IPDPS)*, 2000.
7. Timothy A. Davis, Patrick Amestoy, and Iain S. Duff. An approximate minimum degree ordering algorithm. Technical Report TR-94-039, Computer and Information Sciences Department, University of Florida, Gainesville, FL, 1994.
8. Timothy A. Davis and Iain S. Duff. An unsymmetric-pattern multifrontal method for sparse LU factorization. *SIAM Journal on Matrix Analysis and Applications*, 18(1):140–158, January 1997.
9. James W. Demmel, Stanley C. Eisenstat, John R. Gilbert, Xiaoye S. Li, and Joseph W.-H. Liu. A supernodal approach to sparse partial pivoting. *SIAM Journal on Matrix Analysis and Applications*, 20(3):720–755, 1999.
10. Iain S. Duff, A. M. Erisman, and John K. Reid. *Direct Methods for Sparse Matrices*. Oxford University Press, Oxford, UK, 1990.
11. Iain S. Duff and Jacko Koster. On algorithms for permuting large entries to the diagonal of a sparse matrix. Technical Report RAL-TR-1999-030, Rutherford Appleton Laboratory, April 19, 1999.
12. Iain S. Duff and John K. Reid. The multifrontal solution of unsymmetric sets of linear equations. *SIAM Journal on Scientific and Statistical Computing*, 5(3):633–641, 1984.
13. Alan George and Joseph W.-H. Liu. *Computer Solution of Large Sparse Positive Definite Systems*. Prentice-Hall, NJ, 1981.

14. John R. Gilbert and Joseph W.-H. Liu. Elimination structures for unsymmetric sparse LU factors. *SIAM Journal on Matrix Analysis and Applications*, 14(2):334–352, 1993.
15. Anshul Gupta. A high-performance GEPP-based sparse solver. Technical report, IBM T. J. Watson Research Center, Yorktown Heights, NY, 2001. *ftp://ftp.cs.umn.edu/users/kumar/anshul/parco-01.ps*.
16. Anshul Gupta. Fast and effective algorithms for graph partitioning and sparse matrix ordering. *IBM Journal of Research and Development*, 41(1/2):171–183, January/March, 1997.
17. Anshul Gupta. WSMP: Watson sparse matrix package (Part-II: direct solution of general sparse systems). Technical Report RC 21888 (98472), IBM T. J. Watson Research Center, Yorktown Heights, NY, November 20, 2000. *http://www.cs.umn.edu/~agupta/wsmp.html*.
18. Anshul Gupta. Graph partitioning based sparse matrix ordering algorithms for finite-element and optimization problems. In *Proceedings of the Second SIAM Conference on Sparse Matrices*, October 1996.
19. Anshul Gupta and Yanto Muliadi. An experimental comparison of some direct sparse solver packages. Technical Report RC 21862 (98393), IBM T. J. Watson Research Center, Yorktown Heights, NY, October 25, 2000. *ftp://ftp.cs.umn.edu/users/kumar/anshul/solver-compare.ps*.
20. Anshul Gupta and Lexing Ying. Algorithms for finding maximum matchings in bipartite graphs. Technical Report RC 21576 (97320), IBM T. J. Watson Research Center, Yorktown Heights, NY, October 19, 1999.
21. Steven Michael Hadfield. *On the LU Factorization of Sequences of Identically Structured Sparse Matrices within a Distributed Memory Environment*. PhD thesis, University of Florida, Gainsville, FL, 1994.
22. Xiaoye S. Li and James W. Demmel. Making sparse Gaussian elimination scalable by static pivoting. In *Supercomputing '98 Proceedings*, 1998.
23. Xiaoye S. Li and James W. Demmel. A scalable sparse direct solver using static pivoting. In *Proceedings of the Ninth SIAM Conference on Parallel Processing for Scientific Computing*, 1999.
24. Joseph W.-H. Liu. The multifrontal method for sparse matrix solution: Theory and practice. *SIAM Review*, 34:82–109, 1992.
25. Olaf Schenk, Wolfgang Fichtner, and Klaus Gartner. Scalable parallel sparse LU factorization with a dynamical supernode pivoting approach in semiconductor device simulation. Technical Report 2000/10, Integrated Systems Laboratory, Swiss Federal Institute of Technology, Zurich, November 2000.
26. Kai Shen, Tao Yang, and Xiangmin Jiao. S+: Efficient 2D sparse LU factorization on parallel machines. *SIAM Journal on Matrix Analysis and Applications*, To be published.

On Efficient Application of Implicit Runge-Kutta Methods to Large-Scale Systems of Index 1 Differential-Algebraic Equations*

Gennady Yu. Kulikov[1] and Alexandra A. Korneva[1]

Ulyanovsk State University, L. Tolstoy Str. 42, 432700 Ulyanovsk, Russia

Abstract. In the paper we study how to integrate numerically large-scale systems of semi-explicit index 1 differential-algebraic equations by implicit Runge-Kutta methods. In this case, if Newton-type iterations are applied to the discrete problems we need to solve high dimension linear systems with sparse coefficient matrices. Therefore we develop an effective way for packing such matrices of coefficients and derive special Gaussian elimination for parallel factorization of nonzero blocks of the matrices. As a result, we produce a new efficient procedure to solve linear systems arising from application of Newton iterations to the discretizations of large-scale index 1 differential-algebraic equations obtained by implicit Runge-Kutta methods. Numerical tests support theoretical results of the paper.

1 Introduction

In this paper we deal with index 1 differential-algebraic systems of the form

$$x'(t) = g\big(x(t), y(t)\big), \tag{1a}$$

$$y(t) = f\big(x(t), y(t)\big), \tag{1b}$$

$$x(0) = x^0,\ y(0) = y^0. \tag{1c}$$

where $t \in [0, T]$, $x(t) \in \mathbf{R}^m$, $y(t) \in \mathbf{R}^n$, $g : D \subset \mathbf{R}^{m+n} \to \mathbf{R}^m$, $f : D \subset \mathbf{R}^{m+n} \to \mathbf{R}^n$, and where initial conditions (1c) are consistent; i. e., $y^0 = f(x^0, y^0)$. Note that we consider only autonomous systems because any nonautonomous system may be converted to an autonomous one by introducing a new independent variable.

To solve problem (1) numerically, we apply an l-stage implicit Runge-Kutta (RK) method given by the table

$$\begin{array}{c|c} c & A \\ \hline & b^T \end{array},$$

* This work was supported in part by the Russian Foundation of the Basic Research (grants No. 01-01-00066 and No. 00-01-00197).

V.N. Alexandrov et al. (Eds.): ICCS 2001, LNCS 2073, pp. 832–841, 2001.
© Springer-Verlag Berlin Heidelberg 2001

where A is a real matrix of dimension $l \times l$, b and c are real vectors of dimension l, to problem (1) and obtain the following discrete problem:

$$x_{ki} = x_k + \tau \sum_{j=1}^{l} a_{ij} g(x_{kj}, y_{kj}), \tag{2a}$$

$$y_{ki} = f(x_{ki}, y_{ki}), \quad i = 1, 2, ..., l, \tag{2b}$$

$$x_{k+1} = x_k + \tau \sum_{i=1}^{l} b_i g(x_{ki}, y_{ki}), \tag{2c}$$

$$y_{k+1} = f(x_{k+1}, y_{k+1}), \quad k = 0, 1, ..., K-1, \tag{2d}$$

$$x_0 = x^0, \quad y_0 = y^0 \tag{2e}$$

where τ is a stepsize. Algebraic system (2) is solved then by an iterative method. Usually the iterative process is taken in the form of simple or Newton-type iterations with trivial (or nontrivial) predictor [1], [4], [7], [8], [10], [12].

Newton (or modified Newton) iterations are more preferable for solving differential-algebraic equations (1) than simple ones. First, RK method (2) is convergent in the case of simple iterations only if the infinite norm of the Jacobian of the right-hand part of (1b) is bounded on a convex set containing the exact solution of the original problem by a constant $d < 1$ (see theorem 2 in [8] and theorem 3 in [10]). In the case of Newton-type iterations it is enough to satisfy the implicit function theorem for algebraic part (1b) in order to obtain convergent numerical methods (see theorems 3, 4 in [8] and theorems 1, 2 in [10]). The last condition holds obviously for any sufficiently smooth differential-algebraic system (1) of index 1. This follows from [2], [3] and [6]. Second, when using Newton (full or modified) method we can limit ourselves by a less number of iterations to attain the order of the underlying RK formula. Moreover, if we apply an RK method of sufficiently high stage order with appropriate nontrivial predictor to problem (1) then we can carry out only 2 iterations per time point to get the maximum order convergence [10].

The only drawback of Newton iterations is severe requirements on RAM and CPU time caused by the increasing dimension of discrete problem (2) in l times when an l-stage implicit RK method is used. Thus, it is necessary to solve linear systems of dimension $(m+n)l$ many times during the integration. Therefore the basic problem is how to simplify the numerical solving of the linear systems with coefficient matrices of the special form arising from the application of Newton methods (full or modified) to differential-algebraic system (1).

In the paper we use the special structure of the matrices mentioned above to obtain a modification of Gaussian elimination. This modification allows RAM and CPU time to be significantly reduced in the numerical integration of problem (1) by an implicit RK method. We give the estimates of such reduction both in theory and in practice.

Finally, we study how to integrate numerically large-scale systems of differential-algebraic equations of index 1. In this case we need to solve high dimension

linear systems with sparse coefficient matrices. We develop firstly an effective way for packing such matrices of coefficients. After that we derive special Gaussian elimination for parallel factorization of nonzero blocks of the matrices. Thus, we produce a new efficient procedure to solve linear systems arising from application of Newton iterations to the discretizations of large-scale index 1 differential-algebraic systems obtained by implicit RK methods. We also give numerical tests which support the theoretical results.

2 Efficient Implementation of Iterative Runge-Kutta Methods

In the preceding section we have shown that Newton iterations are effective to solve semi-explicit index 1 differential-algebraic equations numerically. The basic part of the Newton method applied to discrete problem (2) consists of solving linear systems of the form

$$\partial \bar{F}_k^\tau(Z_{k+1}^{i-1})(Z_{k+1}^{i-1} - Z_{k+1}^{i}) = \bar{F}_k^\tau Z_{k+1}^{i-1} \tag{3}$$

where lower indices mean time points, and upper ones denote iterations. Here $Z_{k+1} \stackrel{\text{def}}{=} \left((z_{k1})^T, ..., (z_{k,l-1})^T, (z_{kl})^T\right)^T \in \mathbf{R}^{(m+n)l}$, where the vector $z_{kj} \stackrel{\text{def}}{=} \left((x_{kj})^T, (y_{kj})^T\right)^T \in \mathbf{R}^{m+n}$, $j = 1, 2, \ldots, l$, unites components of the j-th stage value of l-stage RK formula (2). The mapping $\bar{F}_k^\tau$ is the nontrivial part of discrete problem (2) for computing the stage values Z_{k+1}, and $\partial \bar{F}_k^\tau(Z_{k+1}^{i-1})$ denotes the Jacobian of the mapping $\bar{F}_k^\tau$ evaluated at the point Z_{k+1}^{i-1}.

It is easy to see that the matrix $\partial \bar{F}_k^\tau(Z_{k+1}^{i-1})$ has the block structure

$$\partial \bar{F}_k^\tau(Z) \stackrel{\text{def}}{=} \begin{pmatrix} \partial \bar{F}_k^\tau(Z)_1 \\ \partial \bar{F}_k^\tau(Z)_2 \\ \vdots \\ \partial \bar{F}_k^\tau(Z)_l \end{pmatrix} \tag{4}$$

where each block $\partial \bar{F}_k^\tau(Z)_j$, $j = 1, 2, \ldots, l$, is an $(m+n) \times (m+n)l$-matrix of the form

$$\begin{pmatrix} O(\tau) & \cdots & O(\tau) & 1+O(\tau) & \cdots & O(\tau) & O(\tau) & \cdots & O(\tau) & O(\tau) & \cdots & O(\tau) \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ O(\tau) & \cdots & O(\tau) & O(\tau) & \cdots & 1+O(\tau) & O(\tau) & \cdots & O(\tau) & O(\tau) & \cdots & O(\tau) \\ 0 & \cdots & 0 & z & \cdots & z & z & \cdots & z & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & z & \cdots & z & z & \cdots & z & 0 & \cdots & 0 \end{pmatrix} \begin{matrix} \} m \\ \\ \\ \} n \end{matrix}$$

$$\underbrace{\qquad}_{(m+n)(j-1)} \quad \underbrace{\qquad\qquad}_{m} \quad \underbrace{\qquad}_{n} \quad \underbrace{\qquad\qquad}_{(m+n)(l-j)}$$

Here z means in general a nontrivial element.

Having used the structure of matrix (4), Kulikov and Thomsen suggested in [7] to exclude the zero blocks situated below the main diagonal from LU-factorization and forward substitution. That reduces the number of arithmetical operations and, hence, CPU time significantly when linear system (3) is solved. However, the advantage of the new version of Gaussian elimination becomes greater if we have interchanged the x-components of the vector Z_{k+1} with the corresponding y-components. It means that we have interchanged the first m rows with the last n ones in each submatrix $\partial \bar{F}_k^T(Z)_j$. In this case we can exclude all the zero blocks of matrix (4) from the LU-factorization, forward and backward substitutions and solve problem (3) very effectively [9].

Let us now compare efficiency of all the versions of Gaussian elimination for linear system (3). To do this, we compute and compare the total number of multiplications and divisions for each version when $m = n = 2$ and $l = 1, 2, 3, 4$.

It is well-known that the ordinary Gauss method requires

$$\frac{(m+n)l\Big(\big((m+n)l\big)^2 + 3(m+n)l - 1\Big)}{3} \tag{5}$$

operations of multiplication and division (see, for example, [13]). The total number of operations for Kulikov and Thomsen's version (*Modification I*) can be easily obtained from [7]

$$\begin{aligned}\big((m+n)l\big)^2 &+ \frac{\big((m+n)l-1\big)(m+n)l\big(2(m+n)l-1\big)}{6}\\ &+ \frac{(m+n)l\big(ml+n-1\big)}{2} - \frac{n(m+n)^2(l-1)l(2l-1)}{6}\\ &- \frac{n(m+n)(m+n-1)l(l-1)}{4}.\end{aligned} \tag{6}$$

The formula

$$\begin{aligned}(m+n)l(ml+n) &+ \frac{(m+n-1)(m+n)\big(2(m+n)-1\big)l}{6}\\ &+ \frac{m(m+n)(2m+n)l(l-1)}{4} + \frac{m^2(m+n)(l-1)l(2l-1)}{6}\\ &+ \frac{n(n-1)l}{2} + \frac{m(m-1)l}{2} + mnl\end{aligned} \tag{7}$$

gives the number of multiplications and divisions for Kulikov and Korneva's version (*Modification II*) [9].

Now we substitute $m = n = 2$ and $l = 1, 2, 3, 4$ in formulas (5)–(7) and obtain Table 1. We see from the table that Modification II is the most preferable to be used even in such a low dimension case. For example, this version of Gaussian

Table 1. Total numbers of multiplications and divisions when $m = n = 2$

Number of stages, l	Gauss method	Modification I	Modification II
1	36	36	36
2	232	180	128
3	716	496	308
4	1616	1048	608

Table 2. RAM (in bytes) needed to store matrix (4) when $m = n = 2$

Number of stages, l	Gauss method	Modification I	Modification II
1	128	128	128
2	512	448	384
3	1152	960	768
4	2048	1664	1280

elimination requires roughly three times less operations for 4-stage implicit RK formulas than the ordinary Gauss method. It is also better than Modification I. The advantage of this method will obviously increase for a higher dimension differential-algebraic systems (1).

Except CPU time, it is important to compare RAM needed to store matrix (4). If we assume that the type *double* is used to store elements of the matrix then Table 2 gives such information. These data also confirm advantage of Modification II over other methods.

3 Large-Scale Systems

In Section 2 we presented the efficient way of applying implicit RK methods to problem (1). Unfortunately, it is not enough when we solve large-scale semi-explicit index 1 differential-algebraic systems. As an example, we may take the model of overall regulation of body fluids [5]. This model is a problem of the form (1) containing about two hundred variables. Having applied any implicit 3- or 4-stage RK method to the model we encounter the situation when the dimension of discrete problem (2) is too high to solve it by Newton iterations. On the other hand, the Jacobian of a large-scale differential-algebraic system is often a sparse matrix. For instance, the model mentioned above is the case. Thus, we must solve two problems in this section. The first problem is how to pack a matrix of the form (4) excluding all trivial elements. The second one is how to implement Modification II of Gaussian elimination for the packed matrix (4) efficiently.

First of all we rearrange the variables and define the vectors:

$$X_{k+1} \stackrel{\text{def}}{=} \left((x_{k1})^T, ..., (x_{k,l-1})^T, (x_{kl})^T\right)^T \in \mathbf{R}^{ml},$$

$$Y_{k+1} \stackrel{\text{def}}{=} \left((y_{k1})^T, ..., (y_{k,l-1})^T, (y_{kl})^T\right)^T \in \mathbf{R}^{nl}.$$

Now $Z_{k+1} \stackrel{\text{def}}{=} \left((X_{k+1})^T, (Y_{k+1})^T\right)^T$ and matrix (4) has the form

$$\partial \bar{F}_k^\tau(Z) \stackrel{\text{def}}{=} \begin{pmatrix} \partial \bar{F}_k^\tau(Z)^Y \\ \partial \bar{F}_k^\tau(Z)^X \end{pmatrix}. \tag{8}$$

Here each submatrix has also the block structure

$$\partial \bar{F}_k^\tau(Z)^Y \stackrel{\text{def}}{=} \begin{pmatrix} \partial \bar{F}_k^\tau(Z)_1^Y \\ \partial \bar{F}_k^\tau(Z)_2^Y \\ \vdots \\ \partial \bar{F}_k^\tau(Z)_l^Y \end{pmatrix} \quad \text{and} \quad \partial \bar{F}_k^\tau(Z)^X \stackrel{\text{def}}{=} \begin{pmatrix} \partial \bar{F}_k^\tau(Z)_1^X \\ \partial \bar{F}_k^\tau(Z)_2^X \\ \vdots \\ \partial \bar{F}_k^\tau(Z)_l^X \end{pmatrix}$$

where

$$\partial \bar{F}_k^\tau(Z)_i^Y \stackrel{\text{def}}{=} \underbrace{\begin{pmatrix} 0 & \cdots & 0 & z & \cdots & z & 0 & \cdots & 0 & z & \cdots & z & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & z & \cdots & z & 0 & \cdots & 0 & z & \cdots & z & 0 & \cdots & 0 \end{pmatrix}}_{\begin{matrix} n(i-1) & n & n(l-i)+m(i-1) & m & m(l-i) \end{matrix}} \Bigg\} n$$

and

$$\partial \bar{F}_k^\tau(Z)_i^X \stackrel{\text{def}}{=} \underbrace{\begin{pmatrix} O(\tau) & \cdots & O(\tau) & 1+O(\tau) & \cdots & O(\tau) & O(\tau) & \cdots & O(\tau) \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ O(\tau) & \cdots & O(\tau) & O(\tau) & \cdots & 1+O(\tau) & O(\tau) & \cdots & O(\tau) \end{pmatrix}}_{\begin{matrix} nl+m(i-1) & m & m(l-i) \end{matrix}} \Bigg\} m$$

We note that, when solving linear system (3) with matrix (8) by Modification II of Gaussian elimination, LU-factorization of any submatrix $\partial \bar{F}_k^\tau(Z)_i^Y$ does not influence the submatrices $\partial \bar{F}_k^\tau(Z)_j^Y$ for $j \neq i$. This means that the factorization of the matrix $\partial \bar{F}_k^\tau(Z)^Y$ falls into l independent LU-factorizations of the submatrices $\partial \bar{F}_k^\tau(Z)_i^Y$, $i = 1, 2, \ldots, l$. Moreover, structures of the similar nonzero blocks of all the submatrices $\partial \bar{F}_k^\tau(Z)_i^Y$ (i.e., the number and the places of nonzero elements) coincide, if the stepsize τ is sufficiently small.

Taking into account the above observation we conclude that the storage of nonzero elements of the matrix $\partial \bar{F}_k^\tau(Z)^Y$ has to give simultaneous access to all elements corresponding components of the stage values with the same subscript. It allows for the parallel factorization of the matrix $\partial \bar{F}_k^\tau(Z)^Y$. To provide this, we will store the matrix $\partial \bar{F}_k^\tau(Z)^Y$ in the form of an array of links to chained lists. Every component of the list representation consists of:

– an l dimension array to store elements of the matrix $\partial \bar{F}_k^\tau(Z)^Y$ corresponding components of the stage values with the same subscript (f_{ij});

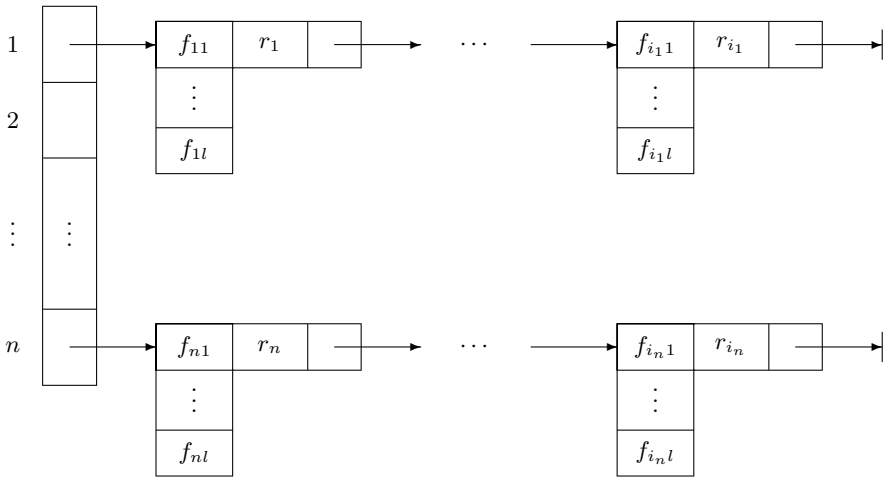


Fig. 1. The storage of nonzero elements of the matrix $\partial \bar{F}_k^\tau(Z)^Y$.

– the subscript of components of the stage values (r_i);
– a link to the next element of the list (or to *nil*).

Since the dimension of the matrix $\partial \bar{F}_k^\tau(Z)^Y$ is equal to nl we need n such lists (see Fig. 1).

To store the matrix $\partial \bar{F}_k^\tau(Z)^X$, we can use any packing appropriate for sparse matrices in a general case because the matrix of coefficients of the RK method may have zero elements and, hence, $\partial \bar{F}_k^\tau(Z)^X$ also contains zero blocks. Fortunately, we will have great advantage by applying only RK methods with *dense coefficient matrix* A (i.e., $a_{ij} \neq 0$, $i, j = 1, 2, \ldots, l$) to problem (1). We call such methods *dense RK methods*. For example, Gauss methods are dense RK methods (see [4]). And there exist no dense methods among explicit RK formulas.

It is easy to see that the matrix $\partial \bar{F}_k^\tau(Z)^X$ can be obtained by pairwise products of elements of the matrices τA and $\partial g^l(Z)$ where the $m \times (m+n)l$-matrix

$$\partial g^l(Z_{k+1}) \stackrel{\text{def}}{=} \big(\partial_{y_{k1}} g(z_{k1}) \ \ldots \ \partial_{y_{kl}} g(z_{kl}) \ \ \partial_{x_{k1}} g(z_{k1}) \ \ldots \ \partial_{x_{kl}} g(z_{kl})\big). \qquad (9)$$

Therefore if we have applied a dense RK formula to problem (1) then all the submatrices $\partial \bar{F}_k^\tau(Z)_i^X$, $i = 1, 2, \ldots, l$, of the matrix $\partial \bar{F}_k^\tau(Z)^X$ have the same structure, maybe, except the diagonal elements. Moreover, the structure of any submatrix $\partial \bar{F}_k^\tau(Z)_i^X$ and the structure of the matrix $\partial g^l(Z)$ coincide with the exception mentioned above. Then all nonzero elements of the matrix $\partial \bar{F}_k^\tau(Z)^X$ can be reconstructed by the matrix τA and nonzero elements of the matrix $\partial g^l(Z)$. Thus, if the stage number $l > 1$, and m, n are large, it is better to store two matrices of dimensions $l \times l$ and $m \times (m+n)l$ instead of one matrix of dimension $ml \times (m+n)l$.

Now we discuss Gaussian elimination for system (3) with matrix (8). It was noted earlier that the elimination of variables from system (3) is splitted up

into two stages. We eliminated the y-components, by using the parallel factorization of the submatrices $\partial \bar{F}_k^\tau(Z)_i^Y$, $i = 1, 2, \ldots, l$, on the first stage. Then we eliminated the x-components. The first stage is more important for optimization because the most part of arithmetical operations falls on it. Therefore we give further a way to decrease the number of operations at this stage.

Let us consider the reduced matrix

$$\begin{pmatrix} \partial \bar{F}_k^\tau(Z)^Y \\ \partial g^l(Z) \end{pmatrix} \tag{10}$$

of dimension $(nl + m) \times (m + n)l$. The next theorem establishes that we can use the reduced matrix (10) instead of the full matrix (8) while eliminating the y-components. That is more preferable for us.

Theorem 1. *Let a dense l-stage RK formula with coefficient matrix A be used for constructing matrix (8). Then the matrix $\left[\partial \bar{F}_k^\tau(Z)^X\right]^{(\mu)}$ obtained after the μ-th step of Gaussian elimination can be reconstructed uniquely by pairwise products of elements of the matrices τA and $\left[\partial g^l(Z)\right]^{(\mu)}$ when $0 \le \mu \le nl$.*

The proof of theorem 1 will appear in [11]. Thus, taking into account this theorem we use the lower dimension matrix (10) on the first nl steps of the Gaussian elimination. After that we reconstruct the matrix $\left[\partial \bar{F}_k^\tau(Z)^X\right]^{(nl)}$ by the matrices τA and $\left[\partial g^l(Z)\right]^{(nl)}$ and proceed the elimination of the x-components of system (3). However, we must remember the parallel factorization of the matrix $\partial \bar{F}_k^\tau(Z)^Y$. By this reason, we have also to store nonzero elements of the matrix $\partial g^l(Z)$ given in (9) by the packing suggested for the matrix $\partial \bar{F}_k^\tau(Z)^Y$ (see Fig. 1). The only difference is the number of the chained lists that are necessary to store the matrix $\partial g^l(Z)$. In this case we use m such lists.

4 Numerical Example

To compare all the versions of Gaussian elimination presented in this paper, we take the model of overall regulation of body fluids mentioned above as a test problem. The version from Section 3 is called further *Modification III*. To solve the problem in the interval $[0, 10]$, we apply Gauss-type implicit RK methods up to order 8 with the stepsize $\tau = 1/60$ and fulfil two Newton iterations per time point. Table 3 contains execution time (in sec.) for all the versions of Gaussian elimination. Lines in the table mean that the Jacobian of the discrete problem exceeds the available RAM.

This practical example shows that Modification III is the best method to solve linear problems arising in application of implicit RK formulas to large-scale systems of semi-explicit index 1 differential-algebraic equations. Indeed, we see from Figure 2 that the Jacobian of the model of overall regulation of body fluids is a sparse matrix (points mean nonzero elements), and Modification III operates only with the nonzero elements. Also it is important to note that the growth of

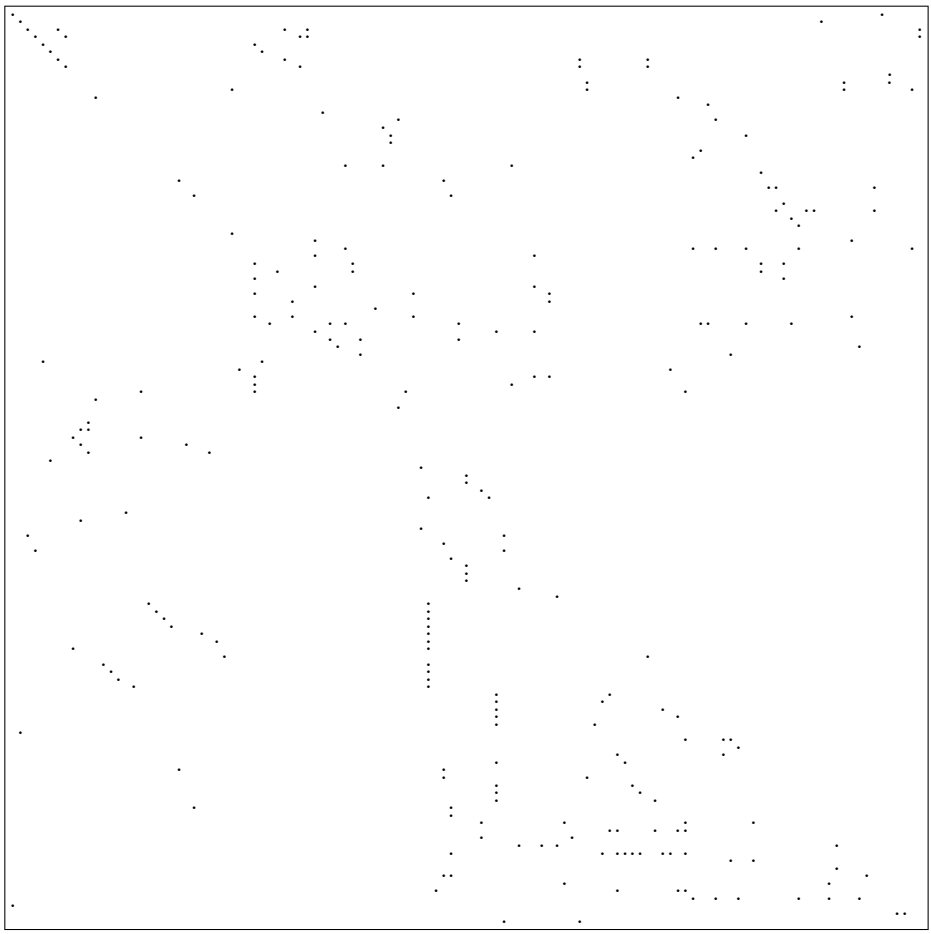


Fig. 2. The structure of the Jacobian of the model of overall regulation of body fluids.

Table 3. Execution time (in sec.) for the processor Intel Pentium 200

Number of stages, l	Gauss method	Modification I	Modification II	Modification III
1	1169.85	993.05	983.77	51.80
2	9271.15	4146.33	2551.29	215.80
3	—	—	—	679.71
4	—	—	—	763.90

the execution time decreases with the growth of the number of stages in implicit RK formulas. It is a good result to apply implicit RK formulas of high order in practice.

References

1. Ascher, U.M., Petzold, L.P.: Computer methods for ordinary differential equations and differential-algebraic equations. SIAM, Philadelphia, 1998
2. Gear, C.W., Petzold, L.R.: ODE methods for the solution of differential/algebraic systems. SIAM J. Numer. Anal. **21** (1984) 716–728
3. Gear, C.W.: Differential-algebraic equations index transformations. SIAM J. Sci. Stat. Comput., **9** (1988) 39–47
4. Hairer, E., Wanner, G.: Solving ordinary differential equations II: Stiff and differential-algebraic problems. Springer-Verlag, Berlin, 1991
5. Ikeda, N., Marumo, F., Shiratare, M., Sato, T.: A model of overall regulation of body fluids. Ann. Biomed. Eng. **7** (1979) 135–166
6. Kulikov, G.Yu.: The numerical solution of the autonomous Cauchy problem with an algebraic relation between the phase variables (non-degenerate case). (*in Russian*) Vestnik Moskov. Univ. Ser. 1 Mat. Mekh. (1993) No. 3, 6–10; *translation in* Moscow Univ. Math. Bull. **48** (1993) No. 3, 8–12
7. Kulikov, G.Yu., Thomsen, P.G.: Convergence and implementation of implicit Runge-Kutta methods for DAEs. Technical report 7/1996, IMM, Technical University of Denmark, Lyngby, 1996
8. Kulikov, G.Yu.: Convergence theorems for iterative Runge-Kutta methods with a constant integration step. (*in Russian*) Zh. Vychisl. Mat. Mat. Fiz. **36** (1996) No. 8, 73–89; *translation in* Comp. Maths Math. Phys. **36** (1996) No. 8, 1041–1054
9. Kulikov, G.Yu., Korneva, A.A.: On effective implementation of iterative Runge-Kutta methods for differential-algebraic equations of index 1. (*in Russian*) In: Basic problems of mathematics and mechanics. **3** (1997), Ulyanovsk State University, Ulyanovsk, 103–112
10. Kulikov, G.Yu.: Numerical solution of the Cauchy problem for a system of differential-algebraic equations with the use of implicit Runge-Kutta methods with nontrivial predictor. (*in Russian*) Zh. Vychisl. Mat. Mat. Fiz. **38** (1998) No. 1, 68–84; *translation in* Comp. Maths Math. Phys. **38** (1998) No. 1, 64–80
11. Kulikov, G.Yu., Korneva, A.A.: On numerical solution of large-scale systems of index 1 differential-algebraic equations. (*in Russian*) Fundam. Prikl. Mat. (to appear)
12. Kværnø, A.: The order of Runge-Kutta methods applied to semi-explicit DAEs of index 1, using Newton-type iterations to compute the internal stage values. Technical report 2/1992, Mathematical Sciences Div., Norwegian Institute of Technology, Trondheim, 1992
13. Samarskiy, A.A., Gulin, A.V.: Numerical methods. Nauka, Moscow, 1989

On the Efficiency of Nearest Neighbor Searching with Data Clustered in Lower Dimensions

Songrit Maneewongvatana and David M. Mount
{songrit,mount}@cs.umd.edu

Department of Computer Science
University of Maryland
College Park, Maryland

1 Introduction

Nearest neighbor searching is an important and fundamental problem in the field of geometric data structures. Given a set S of n *data points* in real d-dimensional space, $\mathbf{R}^d$, we wish to preprocess these points so that, given any *query point* $q \in \mathbf{R}^d$, the data point nearest to q can be reported quickly. We assume that distances are measured using any Minkowski distance metric, including the Euclidean, Manhattan, and max metrics. Nearest neighbor searching has numerous applications in diverse areas of science.

In spite a recent theoretical progress on this problem, the most popular linear-space data structures for nearest neighbor searching are those based on hierarchical decompositions of space. Although these algorithms do not achieve the best asymptotic performance, they are easy to implement, and can achieve fairly good performance in moderately high dimensions. Friedman, Bentley, and Finkel [FBF77] showed that kd-trees achieve $O(\log n)$ expected-case search time and $O(n)$ space, for fixed d, assuming data distributions of bounded density. Arya, et al. [AMN+98] showed that $(1+\epsilon)$ approximate nearest neighbor queries can be answered $O((d/\epsilon)^d \log n)$ time, assuming $O(dn)$ storage. There have been many approaches to reduce the exponential dependence on d [IM98,Kle97].

The unpleasant exponential factors of d in the worst-case analyses of some data structures would lead one to believe that they would be unacceptably slow, even for moderate dimensional nearest neighbor searching ($d < 20$). Nonetheless, practical experience shows that, if carefully implemented, they can applied successfully to problems in these and higher dimensions [AMN+98].

The purpose of this paper is to attempt to provide some theoretical explanation for a possible source for this unexpectedly good performance, and to comment on the limitations of this performance. Conventional wisdom holds that because of dependencies between the dimensions, high dimensional data sets often consist of many low-dimensional clusters. A great deal of work in multivariate data analysis deals with the problems of dimension reduction and determining the *intrinsic dimensionality* of a data set [CP96]. For example, this may be done through the use of techniques such as the Karhunen-Loeve transform [Fuk90].

This suggests the question of how well do data structures take advantage of the presence of low-dimensional clustering in the data set to improve the search?

V.N. Alexandrov et al. (Eds.): ICCS 2001, LNCS 2073, pp. 842–851, 2001.
© Springer-Verlag Berlin Heidelberg 2001

Traditional worst-case analysis does not model the behavior of data structures in the presence of simplifying structure in the data. In fact, it focuses on worst-case situations, which may be rare in practice. Even expected-case analyses based on the assumption of uniformly distributed data [FBF77,Cle79] are not dealing with "easy" instances since the curse of dimensionality is felt in its full force.

We consider the following very simple scenario. Assuming that the data points and query points are sampled uniformly from a k-dimensional hyperplane (or k-flat), where $k < d$, what is the expected-case search time for kd-trees as a function of n, k and d? In [FBF77] it is shown that when $k = d$ and if boundary effects (explained in [AMN96]) are ignored, the expected number of leaf cells in the tree to be visited is at most $(G(d)^{1/d} + 1)^d$, where $G(d)$ is the ratio of the volumes of a d-dimensional hypercube and a maximal enclosed ball for the metric inside the hypercube. These results rely on the fact that when data points are uniformly distributed, the cells of the kd-tree can be approximated by d-dimensional hypercubes. However this is not the case when data points lie on a lower dimensional hyperplane.

It is natural to conjecture that if $k \ll d$, then search times grow exponentially in k but not in d. Indeed, we show that this is the case, for a suitable variant of the kd-tree. We introduce a new splitting method, called the *canonical sliding-midpoint splitting method*. This is a variant of a simpler splitting method called *sliding-midpoint*, which is implemented in the ANN approximate nearest neighbor library [MA97]. (Definitions are given in the next section.)

Our main result is that canonical sliding-midpoint kd-trees can achieve query times depending exponentially on the intrinsic dimension of data, and not on the dimension of the space. We show that if the data points are uniformly distributed on a k-flat, then the expected number of leaf cells that intersect a nearest neighbor ball is $O(d^{k+2})$. Further, we show that if the points are clustered along a k-flat that is aligned with the coordinate axes, even better performance is possible. The expected number of leaf cells intersecting the nearest neighbor ball decreases to $O((d-k+1)c^k)$, where c is the quantity $(G(k)^{1/k}+1)$. The restrictions of using the canonical sliding-midpoint splitting method and having points lie on a flat do not seem to be easy to eliminate. It is not hard to show that if points are perturbed away from the flat, or if some other splitting method is used, there exist point configurations for which 2^d cells will be visited.

The problem of how hierarchical decomposition methods perform when given data with low intrinsic dimensionality has been studied before. Faloutsos and Kamel [FK94] have shown that under certain assumptions, the query time of range queries in an R-tree depends on the fractal dimension of the data set. Their results do not apply to nearest neighbor queries, because their analysis holds in the limit for a fixed query range as the data size tends to infinity.

We also present empirical results that support our results. Furthermore, we consider its robustness to violations in our assumptions. We consider the cases where there is more than just a single cluster of points, but a number of clusters of points lying on different hyperplanes, and where the points do not lie exactly on the hyperplane, but are subject to small perturbations. These empirical results

bear out the fact that the query times are much more strongly dependent on k than on d.

2 Background

First we recall the basic facts about kd-trees [Ben75]. Consider a set S of n data points in $\mathbf{R}^d$. A kd-tree is a binary tree that represents a hierarchical subdivision of space, using splitting planes that are orthogonal to the coordinate axes. Each node of the kd-tree is associated with a closed rectangular region, called a *cell*. The root's cell is associated with a bounding hypercube that contains all the points of S. Information about splitting dimension and splitting value is associated with each cell. These define an axis-orthogonal *splitting hyperplane*. The points of the cell are partitioned to one side or the other of this hyperplane. The resulting subcells are the children of the original cell. This process continues until the number of points is at most one. There are a number of ways of selecting the splitting hyperplane, which we outline below.

Standard split: Proposed in [FBF77], it selected the splitting dimension to be the one for which point set has the maximum *spread* (difference between the maximum and minimum values). The splitting value is chosen to be the median in that dimension. This method is well-known and widely used.

Midpoint split: The splitting hyperplane passes through the center of the cell and bisects the longest side of the cell. If there are many sides of equal length, any may be chosen first, say, the one with the lowest coordinate index. This is just a binary version of the well-known quadtree and octree decompositions.

Observe that the standard splitting rule produces balanced kd-trees with $O(\log n)$ depth. The midpoint tree has the feature that for all cells, the ratio of the longest to shortest side (the *aspect ratio*) is at most 2. (We will sometimes use the term *box* to mean a cell of bounded aspect ratio.) This is not necessarily true for the standard splitting method. As shown in [AMN+98], bounded aspect ratio is important to the efficiency of approximate nearest neighbor searching. Unfortunately, if the data are clustered, it is possible to have many empty cells that contain no data points. This is not uncommon in practice, and may result in trees that have many more than $O(n)$ nodes.

Note that the set of possible splitting planes in midpoint split is determined by the position of the initial bounding hypercube. For example, suppose that the initial bounding box is affinely mapped to a unit hypercube $[0, 1]^d$. The splitting values are all of the form $k/2^i$, for some odd integer k, $1 \leq k < 2^i$. We call any cell which could result from the application of this method a *midpoint box*. The concept of such a *canonical set* of splitting planes will be considered later.

Unfortunately, there does not seem to be a single simple splitting rule that provides us with all the properties one might wish for (linear size, logarithmic depth, bounded aspect ratio, convexity, constant cell complexity). In [AMN+98] the BBD-tree was introduced. This tree uses a combination of two operations,

splitting and shrinking to provide for all of these properties (except for convexity). The BAR-tree [DGK99] provides all of these properties, by using nonorthogonal splitting planes, but the cells may have as many as 2^d bounding faces.

We now discuss two other splitting methods, the sliding-midpoint and the canonical sliding-midpoint methods. The sliding-midpoint method was first introduced in [MA97] and was subsequently analyzed empirically in [MM99a]. This method produces no empty nodes. Although cells may not have bounded aspect ratio, observe that every skinny cell that is produced by sliding is adjacent to a fat leaf cell. In [MM99b] we show that this is sufficient to satisfy the necessary packing constraint that fat subdivisions possess. The canonical sliding-midpoint method is introduced primarily for technical reasons. The proof of the main theorem of Section 3 relies on the presence of having a canonical set of splitting planes, while retaining the property that no empty cells are produced.

Sliding-midpoint: It first attempts to perform a midpoint split, by considering a hyperplane passing through the center of the cell and bisecting the cell's longest side. If the data points lie on both sides of the splitting plane then the splitting plane remains here. However, if a *trivial split* were to result (in which all the data points lie to one side of the splitting plane), then it "slides" the splitting plane towards the data points until it encounters the first such point. One child is a leaf cell containing this single point, and the algorithm recurses on the remaining points.

Canonical sliding-midpoint: Define the *enclosure* for a cell to be the smallest midpoint box that encloses the cell. During the construction phase, each node of the tree is associated both with its cell and the cell's enclosure. We first try to split the cell using a hyperplane that bisects the longest side of this enclosure (rather than the cell itself). Again, if this results in a trivial split, then it slides the splitting plane towards the data points until it encounters the first such point.

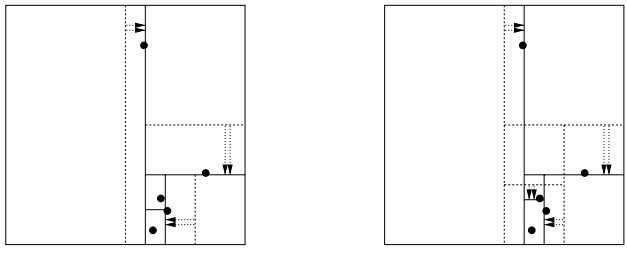


Fig. 1. Sliding-midpoint and canonical sliding-midpoint.

The differences between these two splitting methods is illustrated in Fig. 1. Notice that in the sliding-midpoint method the slides originate from a line that

bisects the cell (shown in dashed lines), whereas in the canonical sliding-midpoint method, the slides originate from the midpoint cuts of the enclosing midpoint cell (shown in dashed lines).

Because of prior sliding operations, the initial split used in the canonical sliding-midpoint method may not pass through the midpoint of the cell. After splitting, the enclosures for the two child cells must also be computed. This can be done in $O(d)$ time [BET93]. Thus, this tree can be constructed in $O(dn \log n)$ time, and has $O(n)$ nodes. just like the sliding-midpoint split kd-tree.

3 Points Clustered on Arbitrarily Oriented Flats

Let F be an arbitrary k-dimensional hyperplane (or k-*flat*, for short) in $\mathbf{R}^d$. We assume that F is in general position, and in particular that F is not parallel to any of the coordinate axes. Let S denote a set of data points sampled from a closed convex, *sampling region* of F according to some probability distribution function. We assume that the distribution function satisfies the following *bounded density assumption* [BWY80]. There exist constants $0 < c_1 \leq c_2$, such that for any convex open subregion of the sampling region with k-dimensional volume V, the probability that a given sampled point lies within this region is in the interval $[c_1 V, c_2 V]$. (This is just a generalization of a uniform distribution but allows some variation in the probability density.)

To avoid having to deal with boundary effects, we will assume that there are sufficiently many data points sampled, and that the query points are chosen from a sufficiently central region, such that with high probability the nearest neighbor ball for any query point lies entirely within the sampling region. More formally, fix any compact convex region on F, called the *query region*, from which query points will be sampled. Let w denote the diameter of this region. Now, take the data points to be sampled from a hypercube of side length $w' > w$ centered around this region, such that the local density of the distribution is independent of w'. Our results hold in the limit as w' tends to infinity. In [AMN96], it is shown that consideration of boundary effects for kd-trees with uniformly distributed points only tends to decrease the number of cells of the tree visited.

Let $B(r)$ denote a ball of radius r. Let $V_F(q, r)$ denote the k-dimensional volume of intersection of F and ball $B(r)$ centered at point q. If we restrict q to lying on F, then $V_F(q, r)$ is a constant for all q, which we denote as $V_F(r)$. Following the approach taken in [AMN96], let us first scale space so that the lower density bound becomes $c_1 = 1/V_k(1)$. After this scaling, a ball of unit radius is expected to contain at least one point of the sample. As observed in [AMN96], as k increases, a ball of unit radius is a very good approximation to the expected nearest neighbor ball. The reason is that $V_F(r)$ is growing as r^k, and so for large k, the probability that a data point lies in $B((1-\delta)r)$ drops rapidly with δ, and the probability that there is at least one point in $B((1+\delta)r)$ increases rapidly with δ.

Consider a kd-tree built for such a distribution, assuming the canonical sliding-midpoint splitting method. Our analysis will focus on the number of leaf

cells of the kd-tree that are visited in the search. The running time of nearest neighbor search (assuming priority search [AMN+98]) is more aptly bounded by the product of the depth of the tree and the time to access these nodes. This access time can be assumed to be $O(\log n)$ either because the tree is balanced, or auxiliary data structures are used. We focus just on the number of leaf cells primarily because in higher dimensions this seems to be the more important factor influencing the running time.

The main result of this section is that the expected number of cells of a canonical sliding-midpoint kd-tree that intersect a unit ball centered on F is exponential in k, but not in d. To see that the proof is nontrivial, suppose that we had stored the points in a regular grid instead. If the nearest neighbor ball contained even a single vertex of the grid, then it would overlap at least 2^d cells. The proof shows that in the canonical midpoint-split kd-tree, it is not possible to generate a vertex that is incident to such a large number of cells when the points lie on a lower dimensional flat. This feature seems to be an important reason that these trees adapt well to the intrinsic dimensionality of the point set. Although it is not clear how to establish this property for other splitting methods in the worst case, we believe that something analogous to this holds in the expected case.

Theorem 1. *Let S be a set of points from $\mathbf{R}^d$ sampled independently from a k-flat F by a distribution satisfying the bounded density assumptions and scaled as described above. Let T be a kd-tree built for S using the canonical sliding-midpoint splitting method. Then, the expected number of leaf cells of T that intersect a unit ball centered on F is $O(d^{k+2})$.*

For the complete proof, see [MM01]. Using Theorem 1 and the observation made earlier that a ball of unit radius is good approximation to (or larger than) the nearest neighbor ball, we have the following bound.

Corollary 1. *The expected number of leaf cells of T encountered in nearest neighbor searching is $O(d^{k+2})$.*

4 Points Clustered on Axis-Aligned Flats

We now consider the case where the set S of data points in $\mathbf{R}^d$ sampled independently from a distribution of bounded density along an axis-aligned k-flat. If in the kd-tree construction we split orthogonal to any of the $d - k$ coordinate axes that are orthogonally to the flat, the points will all lie to one side of this splitting hyperplane. The splitting hyperplane will slide until it lies on the flat. After any sequence of $2(d - k)$ such slides, the flat will be tightly enclosed within a cell. Splits along other axes will be orthogonal to the flat, and so will behave essentially the same a sliding-midpoint decomposition in k-space. The main complication is that the algorithm does not know the location of the flat, and hence these two types of splits may occur in an unpredictable order.

Let $G(k)$ denote the dimension dependent ratio of the volumes of a k-dimensional hypercube and a maximal enclosed k-ball for the metric inside the hypercube. Let $c(k) = (G(k)^{1/k} + 1)$. For example, for the L_∞ (max) metric the metric ball is a hypercube, and $c(k) = 2$. For the L_2 (Euclidean) metric $G(k) = k\Gamma(k/2)/(2^{k+1}\pi^{k/2})$. The proof is presented in [MM01].

Theorem 2. *Let S be a set of points from $\mathbf{R}^d$ sampled independently from an axis-aligned k-flat F by a distribution satisfying the bounded density assumptions described in Section 3. Let T be a kd-tree built for S using the canonical sliding-midpoint splitting method. Then, the expected number of leaf cells of T that intersect a unit ball centered on F is $O((d-k+1)c(k)^k)$.*

5 Empirical Results

We conducted experiments on the query performance of the kd-tree for data sets lying on a lower dimensional flat. We used the ANN library [MA97] to implement the kd-tree. We used priority search to answer queries. We present the total number of nodes, and the number of leaf nodes in our grades, because these parameters are machine-independent and closely correlated with CPU time.

5.1 Distributions Tested

Before discussing what we did in the experiments, we briefly describe the distributions used.

Uniform-on-orthogonal-flat: The dimension of the flat, k, is provided, and k dimensions are chosen at random. Among these dimensions, the points are distributed uniformly over $[-1, 1]$. For the other $(d-k)$ dimensions, we generate a uniform random coordinate that is common to all the points.

Uniform-on-rotated-flat: This distribution is the result of applying r random rotation transformations to the points in uniform-on-orthogonal-flat distribution. In the experiments, r is fixed at $d^2/2$. The flat is therefore rotated in a random direction. Each rotation is through a uniformly distributed angle in the range $[-\pi/2, \pi/2]$ with respect to two randomly chosen dimensions.

Our theoretical results for arbitrary flats apply only to the canonical sliding-midpoint method. This was largely for technical reasons. A natural question is how much this method differs from the more natural sliding-midpoint method. We tested both splitting methods for some other distributions, and discovered that their performances were quite similar. These results as well as additional experiments will be presented in the full version of the paper [MM01].

5.2 Points on a k-Flat

To support our theoretical bounds on number of leaf nodes visited when the point set is on a k-flat, we set up an experiment with both k and d varying,

while fixing the other parameters. This allows us to observe the dependency of the query performance (in terms of the number of nodes visited) relative to d and k. The Uniform-on-orthogonal-flat and Uniform-on-rotated-flat distributions were used in the experiments. We fixed d at 4, 8, 12, 16, 20, 24, 32, 40 (note that the scale is nonlinear), and k ranged from 4 to $\min(d, 16)$. The number of points, n ranged from 40 to 163,840. The queries were sampled from the same distribution. The number of query points was set to $\min(n, 2560)$.

Normally, ANN places a tight bounding box around the points. Such a bounding box would tightly wrap itself around the flat reducing the problem to a purely k-dimensional subdivision. In order to observe the behavior of the scenario considered in Theorem 2, we modified the library so that the initial bounding box is the hypercube $[-1, 1]^d$. The results of this modification are showed in Fig. 2. Note that we plotted the logarithm base 10 of the number of nodes visited. As predicted, the running shows a strong dependence on k, and very little dependence on d. However, it does not grow as fast as predicted by Theorem 2. This suggests that the average case is much better than our theoretical bounds.

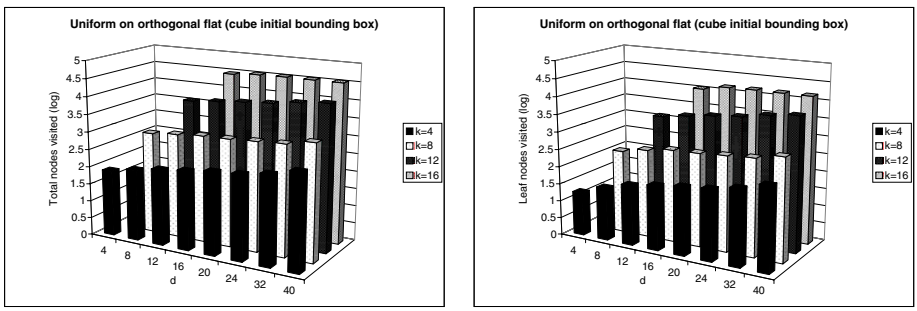


Fig. 2. Number of total and leaf nodes visited, $n = 163,840$, Uniform-on-orthogonal-flat distribution with cube initial bounding box

The Uniform-on-rotated-flat distribution is also used in the experiment to see the effect assuming that data is uniform on an arbitrarily oriented flat. For this distribution, the canonical sliding-midpoint is a little slower (typically, the difference is less than 5%) than the sliding-midpoint in few cases. In general, the number of nodes visited still shows a greater dependence on k than on d, but the dependence on d has increased, as predicted by Theorem 1. Yet, the growth rate is still less than what the theorem predicts. We tested the sensitivity of our result to the presence of multiple clusters. We also ran experiments on the standard kd-tree. Although we could not prove bounds on the expected query time, the empirical performance was quite similar to these other methods. This supports the rule-of-thumb that the standard-split kd-tree tends to perform well when data and query points are chosen from a common distribution.

5.3 Comparison with Theoretical Results

In this section, we take a closer look on whether our theoretical bounds can predict the actual query performance in terms of the number of leaf nodes visited. From Corollary 1, the expected number of leaf nodes of a kd-tree encountered in the search is $O(d^{k+2})$. We model this bound as $L = c_1(c_2 d)^{c_3 k}$, where L is the number of leaf nodes visited and c_1, c_2, c_3 are constants. We set up the experiment such that the data and query distributions are uniform-on-rotated-flat. The parameters are slightly different from the previous experiments. The number of random rotations is d^2, and there is no gaussian noise. The number of data points, n, remains at 163,840. We gathered results for $k = 1$ to 12 and $d = 10, 20, 40, 80$. The results are plotted in Fig 3.

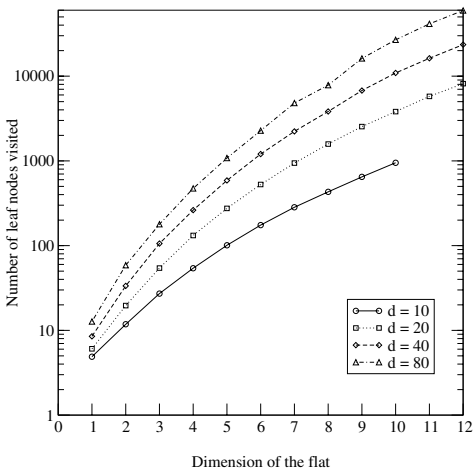


Fig. 3. Number of leaf nodes visited, $n = 163,840$, Uniform-on-rotated-flat distribution

The model suggests that the curves in Fig 3 should be linear. However, the empirical results show that it is not the case. We conjecture that this is due to boundary effects, which would presumably diminish as n increases. These boundary effects are more pronounced for larger values of k [AMN96]. Because of memory limitation, we cannot scale n exponentially with the value of k.

We observed that for smaller values of k (e.g. $k = 1, 2, 3$), the number of leaf nodes visited, L, is almost unchanged when n is increased. It indicates the boundary effects are minimum. Therefore we use the results from $k = 1, 2$ to find values of c_1, c_2, c_3 of our model equation. This yields the following equation,

$$L = 2.054(1.674 \cdot d)^{(0.312 \cdot k)}.$$

References

[AMN96] S. Arya, D. M. Mount, and O. Narayan. Accounting for boundary effects in nearest neighbor searching. *Discrete Comput. Geom.*, 16(2):155–176, 1996.

[AMN+98] S. Arya, D. M. Mount, N. S. Netanyahu, R. Silverman, and A. Wu. An optimal algorithm for approximate nearest neighbor searching. *Journal of the ACM*, 45:891–923, 1998.

[Ben75] J. L. Bentley. Multidimensional binary search trees used for associative searching. *Communications of the ACM*, 18(9):509–517, 1975.

[BET93] M. Bern, D. Eppstein, and S.-H. Teng. Parallel construction of quadtrees and quality triangulations. In *Proc. 3rd Workshop Algorithms Data Struct.*, volume 709 of *Lecture Notes in Computer Science*, pages 188–199. Springer-Verlag, 1993.

[BWY80] J. L. Bentley, B. W. Weide, and A. C. Yao. Optimal expected-time algorithms for closest-point problems. *ACM Trans. Math. Software*, 6(4):563–580, 1980.

[Cle79] J. G. Cleary. Analysis of an algorithm for finding nearest neighbors in euclidean space. *ACM Trans. Math. Software*, 5(2):183–192, 1979.

[CP96] M. Carreira-Perpiñán. A review of dimension reduction techniques. Technical Report CS–96–09, Dept. of Computer Science, University of Sheffield, UK, 1996.

[DGK99] C. Duncan, M. Goodrich, and S. Kobourov. Balanced aspect ratio trees: Combining the advantages of k-d trees and octrees. In *Proc. 10th ACM-SIAM Sympos. Discrete Algorithms*, pages 300–309, 1999.

[FBF77] J. H. Friedman, J. L. Bentley, and R. A. Finkel. An algorithm for finding best matches in logarithmic expected time. *ACM Trans. Math. Software*, 3(3):209–226, 1977.

[FK94] Christos Faloutsos and Ibrahim Kamel. Beyond uniformity and independence: Analysis of R-trees using the concept of fractal dimension. In *Proc. Annu. ACM Sympos. Principles Database Syst.*, pages 4–13, 1994.

[Fuk90] K. Fukunaga. *Introduction to Statistical Pattern Recognition.* Academic Press, 2nd edition, 1990.

[IM98] P. Indyk and R. Motwani. Approximate nearest neighbors: Towards removing the curse of dimensionality. In *Proc. 30th Annu. ACM Sympos. Theory Comput.*, pages 604–613, 1998.

[Kle97] J. M. Kleinberg. Two algorithms for nearest-neighbor search in high dimension. In *Proc. 29th Annu. ACM Sympos. Theory Comput.*, pages 599–608, 1997.

[MA97] D. M. Mount and S. Arya. ANN: A library for approximate nearest neighbor searching. Center for Geometric Computing 2nd Annual Workshop on Computational Geometry, 1997.

[MM99a] S. Maneewongvatana and D. Mount. Analysis of approximate nearest neighbor searching with clustered point sets. In *ALENEX*, 1999.

[MM99b] S. Maneewongvatana and D. Mount. It's okay to be skinny, if your friends are fat. Center for Geometric Computing 4th Annual Workshop on Computational Geometry, 1999.

[MM01] S. Maneewongvatana and D. Mount. On the efficiency of nearest neighbor searching with data clustered in lower dimensions. Technical Report CS-TR-4209, Dept. Computer Science, Univ. Maryland, 2001.

A Spectral Element Method for Oldroyd-B Fluid in a Contraction Channel

Sha Meng, Xin Kai Li, and Gwynne Evans

Institute of Simulation Sciences,
Faculty of Computing Science and Engineering,
De Montfort University, Leicester LE1 9BH, England
smeng@dmu.ac.uk, xkl@dmu.ac.uk, gaevans@dmu.ac.uk
http://www.cse.dmu.ac.uk/ISS/

Abstract. A spectral element method coupled with the EVSS method for computing viscoelastic flows is presented. The nonlinear rheological model, Oldroyd-B, is chosen to simulate the flow of a viscoelastic fluid based on a planar four-to-one abrupt contraction benchmark problem. Numerical results agree well with those in the previous publications.

Keywords: Viscoelastic flow; Spectral element method; Oldroyd-B fluid

1 Introduction

Non-Newtonian fluids, such as multi-grade oils, liquid detergents, polymer melts and molten plastics, are becoming more and more important in many industrial fluids applications. Viscoelastic fluids are non-Newtonian fluids that possess memory. That is, the stress of the fluid depends not only on the stresses actually impressed on them at present, but also on all the stresses to which they have been subjected during their previous deformation history. These fluids are special case of non-Newtonian fluids that lie somewhere in between elastic materials and standard Newtonian fluids. The numerical simulation of such viscoelastic fluids is becoming an effective technique to predict the fluid performance in a wide range of engineering applications.

Most mathematical problems that arise in modeling viscoelastic flows involve the solutions of non-linear partial differential, integro-differential or integral equations. In general, these equations cannot be solved analytically, so numerical methods are required to obtain solutions. The rapid growth in the power and availability of computers has led to the development of many algorithms for solving these equations. Recently, the spectral element method has emerged in the viscoelastic context as a powerful alternative to more traditional methods in predicting flow behaviour in complex fluids. In this paper we mainly focus on the development of an efficient spectral element technique to simulate a viscoelastic flow in a contraction channel.

Contraction flows of viscoelastic fluids are of importance in fundamental flow property measurements as well as in many industrial applications [1]. The theoretical prediction of entry-flow for non-Newtonian fluids still is a difficult task.

V.N. Alexandrov et al. (Eds.): ICCS 2001, LNCS 2073, pp. 852–861, 2001.
© Springer-Verlag Berlin Heidelberg 2001

The difficulty comes from two aspects. One is the constitutive equations that are used to express the relationship between the stress tensor and the velocity gradient and describe the rheological behaviour of viscoelastic fluids which have memory effects and contain nonlinear terms that add to the complexity of the problem; the other one is a geometrical singularity at the re-entrant corner. The research has been dominated by the study of the high Weissenberg numbers and continues to be a benchmark problem in the computational rheology.

In recent years, successful numerical methods have emerged. These include the Hermitian finite element method [7], the 4×4 subelement method [8], the explicitly elliptic momentum equation formulation (EEME) [5], the elastic viscous split stress formulation (EVSS) [10], the consistent streamline upwind Petrov-Galerkin method (SUPG) [4] and the discontinuous Galerkin (DG) method [3]. In this paper, we will present a spectral element formulation to solve the Oldroyd-B viscoelastic flow based on a four-to-one contraction benchmark problem. In section 2, the full set of governing equations for the viscoelastic flow model is presented. The spectral element method is described in section 3, numerical results and discussion are presented in the last section.

2 Mathematical Modeling

The isothermal flow of an incompressible viscoelastic fluid is governed by a set of conservation and constitutive equations. In the absence of body force, the momentum and mass equations can be written as follows

$$\rho\left(\frac{\partial \mathbf{u}}{\partial t}+\mathbf{u}\cdot\nabla\mathbf{u}\right)=-\nabla p+\nabla\cdot\boldsymbol{\tau}, \tag{2.1}$$

$$\nabla\cdot\mathbf{u}=0, \tag{2.2}$$

where ρ is the fluid density, p is the pressure, $\mathbf{u}$ is the velocity vector, and $\boldsymbol{\tau}$ is the extra-stress tensor field. Equations (2.1) and (2.2) must be closed with a constitutive model. In this paper, the Oldroyd-B model is used and defined as

$$\boldsymbol{\tau}+\lambda_1\overset{\nabla}{\boldsymbol{\tau}}=2\eta(\mathbf{D}+\lambda_2\overset{\nabla}{\mathbf{D}}), \tag{2.3}$$

where λ_1 is the relaxation time, λ_2 is the retardation time and η is the shear rate viscosity. $\mathbf{D}$ and $\overset{\nabla}{\boldsymbol{\tau}}$ are the rate of deformation tensor and the upper-convected derivative of the viscoelastic extra-stress, respectively. They are defined as

$$\mathbf{D}=\frac{1}{2}(\nabla\mathbf{u}+(\nabla\mathbf{u})^T),$$

$$\overset{\nabla}{\boldsymbol{\tau}}=\frac{\partial\boldsymbol{\tau}}{\partial t}+\mathbf{u}\cdot\boldsymbol{\nabla}\boldsymbol{\tau}-\boldsymbol{\tau}\cdot(\nabla\mathbf{u})-(\nabla\mathbf{u})^T\cdot\boldsymbol{\tau}.$$

Note that equation (2.3) reduces to the upper-convected Maxwell (UCM) model if $\lambda_2=0$ and to a Newtonian liquid with viscosity η if $\lambda_1=\lambda_2$. The viscoelastic stress tensor can be split into

$$\boldsymbol{\tau}=\boldsymbol{\tau}_1+\boldsymbol{\tau}_2, \tag{2.4}$$

where $\boldsymbol{\tau}_1$ denotes the elastic part of the viscoelastic stress defined as

$$\boldsymbol{\tau}_1 + \lambda \overset{\nabla}{\boldsymbol{\tau}}_1 = 2\eta_1 \mathbf{D},$$

and $\boldsymbol{\tau}_2$ represents the purely viscous component defined as

$$\boldsymbol{\tau}_2 = 2\eta_2 \mathbf{D}.$$

in these equations η_1 is the viscosity of the viscoelastic contribution and η_2 is the viscosity of the Newtonian contribution.

Substituting (2.4) into (2.3), we obtain the Oldroyd-B constitutive equation

$$\boldsymbol{\tau}_1 + \lambda_1 \left(\frac{\partial \boldsymbol{\tau}_1}{\partial t} + \mathbf{u} \cdot \boldsymbol{\nabla} \boldsymbol{\tau}_1 - \boldsymbol{\tau}_1 \cdot (\nabla \mathbf{u}) - (\nabla \mathbf{u})^T \cdot \boldsymbol{\tau}_1 \right) = \eta_1 (\nabla \mathbf{u} + (\nabla \mathbf{u})^T). \quad (2.5)$$

Let $\mathbf{d}$ be an additional unknown

$$\mathbf{d} = \mathbf{D} = \frac{\mathbf{1}}{\mathbf{2}} (\nabla \mathbf{u} + (\nabla \mathbf{u})^{\mathbf{T}}),$$

and replace $\boldsymbol{\tau}_1$ by $\boldsymbol{\tau}$, we obtain $(\mathbf{u}, p, \boldsymbol{\tau}, \mathbf{d})$ in the EVSS formulation

$$\rho \left(\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u} \right) = -\nabla p + \nabla \cdot \boldsymbol{\tau} - 2\eta_1 \nabla \cdot \mathbf{d} + \mathbf{2}\eta \nabla \cdot \mathbf{D}, \quad (2.6)$$

$$\nabla \cdot \mathbf{u} = 0, \quad (2.7)$$

$$\boldsymbol{\tau} + \lambda_1 \overset{\boldsymbol{\nabla}}{\boldsymbol{\tau}} = 2\eta_1 \mathbf{D}, \quad (2.8)$$

$$\mathbf{d} = \mathbf{D}. \quad (2.9)$$

Although we add the same quantity in the right hand side of the momentum equation, the real modification will be appear when we consider different representations for $\mathbf{d}$ and $\mathbf{D}$ in the discrete form of the above system of equations. Furthermore, a dimensionless system of equations can be written as

$$Re \left(\frac{\partial u_i}{\partial t} + u_j \frac{\partial u_i}{\partial x_j} \right) = -\frac{\partial p}{\partial x_j} + \frac{\partial \tau_{ij}}{\partial x_j} - 2(1-\beta) \frac{\partial d_{ij}}{\partial x_j} + \frac{\partial^2 u_i}{\partial x_j^2}, \quad (2.10)$$

$$\frac{\partial u_i}{\partial x_i} = 0, \quad (2.11)$$

$$\tau_{ij} + We \left(\frac{\partial \tau_{ij}}{\partial t} + u_l \frac{\partial \tau_{ij}}{\partial x_l} \right) = (1-\beta) \left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} \right) + We \left(\tau_{il} \frac{\partial u_j}{\partial x_l} + \tau_{jl} \frac{\partial u_i}{\partial x_l} \right), \quad (2.12)$$

$$d_{ij} = \frac{1}{2} \left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} \right), \ \forall\, i, \ \ j, \ \ l = 1, \ \ 2, \quad (2.13)$$

where $Re = \frac{\rho U L}{\eta}$ is the Reynolds number, $We = \frac{\lambda_1 U}{L}$ is the Weissenberg number, and $\beta = \frac{\lambda_2}{\lambda_1}$, which determines the characteristics of the Oldroyd-B fluid.

3 The Spectral Element Discretization

The spectral element method is a high-order weighted-residual technique for partial differential equations that combines the rapid convergence rate of the p-type spectral method with the geometric flexibility of the h-type finite element technique. In the spectral element discretization, the computational domain is broken into macro-spectral elements, and the dependent and independent variables are represented as high-order orthogonal polynomial expansions within the individual subdomains. Variational projection operators and Gauss-Lobatto Legendre numerical quadratures are used to generate the discrete equations, which are then solved by direct or iterative procedures using tensor-product sum-factorization techniques [6].

In order to obtain a weak formulation which is equivalent to the equations $(2.10) - (2.13)$, we introduce the following function spaces:

$$H_0^1(\Omega) = \{\phi : \phi \in H^1(\Omega), \phi = 0 \text{ on } \partial\Omega\},\ L_0^2(\Omega) = \{v : v \in L^2(\Omega), v = 0 \text{ on } \partial\Omega\},$$

where $H^1(\Omega)$ is Soblev space, $L^2(\Omega)$ is the space of square integrable functions. The scalar product can be defined as

$$(\phi, \psi) = \int_\Omega \phi(x)\psi(x)dx,\ \ \forall \phi, \psi \in H^1(\Omega).$$

The spectral element discretization proceeds by breaking up the computational domain Ω into K non-overlapping sub-domains denoted by $\Omega_k, (k = 1, ..., K)$ such that $\Omega = \cup\Omega_k, \forall k, l, k \neq l, \Omega_k \cap \Omega_l = \emptyset$. Each physical element is mapped onto the parent element $\chi^2 = [-1, 1] \times [-1, 1]$, on which a Gauss-Lobatto-Legendre grid is used. We further define

$$X_h = \{u : u|_\Omega \in P_N(\Omega)\} \cap H_0^1(\Omega),\ \ M_h = \{p : p|_\Omega \in P_{N-2}(\Omega)\} \cap L_0^2(\Omega),$$

where $P_N(\Omega)$ denotes the space of all polynomials of degree N or less. It is well known that a choice for the velocity in X_h and the pressure in M_h above avoids spurious pressure nodes and satisfies generalized the Brezzi-Babuska condition [2]. In addition, the second compatibility condition needs to be satisfied for the stress and the rate of deformation tensor spaces. In this paper, we choose $T_h = X_h$ and $D_h = M_h$ in order to have a well-posed solution. Then the spectral element discretization is:
Find $u_{i,h} \in X_h$, $p_h \in M_h$, $\tau_{ij,h} \in T_h$ and $d_{ij,h} \in D_h$ such that

$$\left(\frac{\partial u_{i,h}}{\partial x_j}, \frac{\partial \bar{u}_i}{\partial x_j}\right)_{h,GL} + Re\left(\frac{\partial u_{i,h}}{\partial t}, \bar{u}_i\right)_{h,GL} - \left(p_h, \frac{\partial \bar{u}_i}{\partial x_j}\right)_{h,GL} = \left(\frac{\partial \tau_{ij,h}}{\partial x_j}, \bar{u}_i\right)_{h,GL}$$

$$-2(1-\beta)\left(\frac{\partial d_{ij,h}}{\partial x_j}, \bar{u}_i\right)_{h,GL} - Re\left(u_{j,h}\frac{\partial u_{i,h}}{\partial x_j}, \bar{u}_i\right)_{h,GL}, \tag{3.1}$$

$$\left(\frac{\partial u_{i,h}}{\partial x_i}, q\right)_{h,GL} = 0, \tag{3.2}$$

$$We\left(\frac{\partial \tau_{ij,h}}{\partial t}+u_{l,h}\frac{\partial \tau_{ij,h}}{\partial x_l},\bar{\tau}_{ij}\right)_{h,GL}-We\left(\tau_{il,h}\frac{\partial u_{j,h}}{\partial x_l}+\tau_{jl,h}\frac{\partial u_{i,h}}{\partial x_l},\bar{\tau}_{ij}\right)_{h,GL}$$

$$+\left(\tau_{ij,h},\bar{\tau}_{ij}\right)_{h,GL}=(1-\beta)\left(\frac{\partial u_{i,h}}{\partial x_j}+\frac{\partial u_{j,h}}{\partial x_i},\bar{\tau}_{ij}\right)_{h,GL}, \tag{3.3}$$

$$\left(d_{ij,h},\bar{d}_{ij}\right)_{h,GL}=\frac{1}{2}\left(\frac{\partial u_{i,h}}{\partial x_j}+\frac{\partial u_{j,h}}{\partial x_i},\bar{d}_{ij}\right)_{h,GL}, \tag{3.4}$$

$$\forall \bar{u}_i\in X_h,\ \ \forall q\in M_h,\ \ \forall \bar{\tau}_{ij}\in X_h,\ \ \forall \bar{d}_{ij}\in M_h,\ \ \forall i,\ j,\ l=1,\ 2,$$

where $(*,*)_{h,GL}$ refers to Gauss-Lobatto quadrature which is defined as

$$(f,g)_{h,GL}=\sum_{k=1}^{K}\sum_{m=0}^{M}\sum_{n=0}^{N}\rho_m\rho_n f(\xi_m^k,\phi_n^k)g(\xi_m^k,\phi_n^k)J^k,$$

where ξ_m^k, ϕ_n^k are the locations of the local nodes $\{m;k\}$, $\{n;k\}$ respectively, ξ_m,ϕ_n are the Gauss-Lobatto-Legendre quadrature points, and ρ_m,ϕ_n are the Gauss-Lobatto-Legendre quadrature weights, J^k is the transformation Jacobian on each element.

In this paper we use the Gauss-Lobatto-Legendre polynomials as a basis to span the approximation space X_h and T_h, which is defined as

$$h_i(\xi)=-\frac{1}{N(N+1)L_N(\xi_i)}\frac{(1-\xi^2)L_N^{'}(\xi)}{\xi-\xi_i},\ \ \xi\in[-1,1],\ \ \forall i\in\{0,...,N\},$$

where L_N is the Legendre polynomial of order N, the points ξ_i are the collocation points on the Gauss-Lobatto-Legendre grid. Therefore, the velocity and the stress tensor approximations in the parent element corresponding to element Ω_k are

$$u_h^k(\xi,\phi)=\sum_{p=0}^{M}\sum_{q=0}^{N}u_{pq}^k h_p(\xi)h_q(\phi), \tag{3.5}$$

$$\tau_h^k(\xi,\phi)=\sum_{p=0}^{M}\sum_{q=0}^{N}\tau_{pq}^k h_p(\xi)h_q(\phi), \tag{3.6}$$

where $u_{pq}^k=u(\xi_p^k,\phi_q^k)$, $\tau_{pq}^k=\tau(\xi_p^k,\phi_q^k)$. If we consider the velocity-pressure formulation, it is well known that the mixed interpolations must satisfy a compatibility condition. The framework of the spectral element method [6] has shown that a suitable choice for the pressure approximation space is M_h when the velocity is X_h. Therefore, in this paper, we choose the pressure function in the space M_h and expand it on the interior Gauss-Lobatto-Legendre points as shown in Fig. 1. Thus the pressure approximation can be written as

$$p_h^k(\xi,\phi)=\sum_{p=1}^{M-1}\sum_{q=1}^{N-1}p_{pq}^k\bar{h}_p(\xi)\bar{h}_q(\phi), \tag{3.7}$$

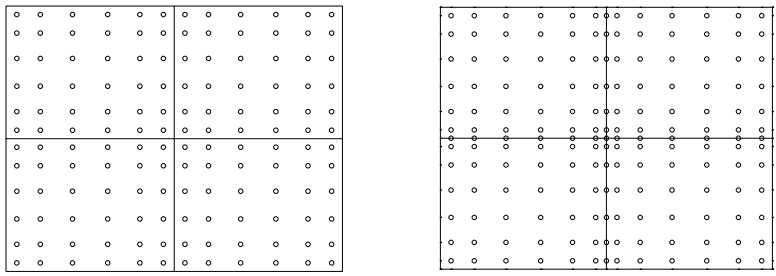


Fig. 1. Spectral element configurations ($K = 4, M = N = 5$). (a) Interior Gauss-Lobatto-Legendre collocation points for the pressure and the deformation tensor. (b) Gauss-Lobatto-Legendre collocation points for the velocity and the stress.

where $p_{pq}^k = p(\xi_p^k, \phi_q^k)$, $\bar{h}_p$ is defined as

$$\bar{h}_p = -\frac{(1-\xi_p^2)L_N^{'}(\xi)}{N(N+1)L_N(\xi_p)(\xi-\xi_p)}, \quad \xi \in [-1,1], \quad \forall p \in \{1,...,N-1\}.$$

Similarly, we define the approximation of the deformation tensor as

$$d_h^k(\xi,\phi) = \sum_{p=1}^{M-1}\sum_{q=1}^{N-1} d_{pq}^k \bar{h}_p(\xi)\bar{h}_q(\phi), \tag{3.8}$$

where $d_{pq}^k = d(\xi_p^k, \phi_q^k)$.

The velocity, pressure, stress and deformation tensor expansions $(3.5) - (3.8)$ are now inserted into equations $(3.1) - (3.4)$ and the discrete equations are generated by choosing appropriate test functions $\bar{\mathbf{u}}$ and $\bar{\boldsymbol{\tau}}$ in X_h whose values at a point (ξ_p, ϕ_q) are unity and zero at all other Gauss-Lobatto-Legendre points, and test functions q and $\bar{\mathbf{d}}$ in M_h whose values are unity at point (ξ_p, ϕ_q) and zero at all other interior Gauss-Lobatto-Legendre points. In this way we obtain the system of algebraic equations

$$\begin{aligned} A\mathbf{u} - B^T p &= f, \\ -B \cdot \mathbf{u} &= 0, \\ C\boldsymbol{\tau} &= g, \\ E\mathbf{d} &= h, \end{aligned}$$

where A is the discrete Helmholtz operator, B is the discrete gradient operator, C is the stress tensor matrix, E is the deformation tensor matrix, f, g, h are the right hand side vectors, which are incorporated with boundary conditions.

4 The Decoupling Algorithm

Now for each time step, the algorithm consists of the following steps: Given an initial approximation $(u_i^0, p^0, \tau_{ij}^0, d_{ij}^0)$,

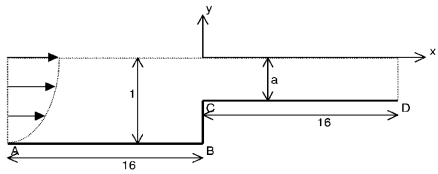


Fig. 2. The four-to-one planar contraction flow geometry.

Step 1: calculate the pressure p^n from the conservation equation by the Uzawa method [6].

Step 2: calculate the velocity $\mathbf{u}^n$ from the momentum equation using the stress $\boldsymbol{\tau}^{n-1}$ obtained from a previous iteration.

Step 3: calculate the stress $\boldsymbol{\tau}^n$ from the constitutive equation using $\mathbf{u}^n$.

Setp 4: calculate the deformation tensor $\mathbf{d}^n$ using the velocity field $\mathbf{u}^n$.

Step 5: check the convergence and return to step 1 if necessary.

5 Numerical Results

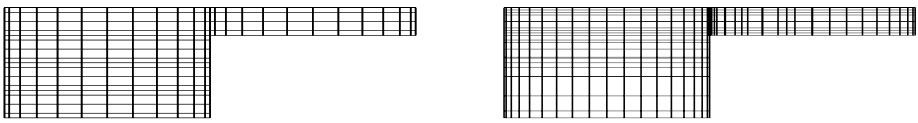
In this section, numerical results are presented for a four-to-one abrupt planar contraction. We adopt the ratio $\beta = \frac{1}{9}$ in order to compare with already published results. The difficulty of the four-to-one planar contraction problem is the existence of a singular solution which is caused by the geometric singularity at the re-entrant corner. The singularity in the viscoelatic flow is stronger than in the Newtonian flow.

Since the geometry is assumed to be symmetric about the central line, we need only consider the lower half of the channel. Fig. 2 shows the flow geometry. The height of the inflow half channel is taken as unity and the height of outflow channel is taken to be $a = \frac{1}{4}$. The length of inflow channel is taken to be 16 as is the length of outflow channel. Define $U = 1$ and $L = 1$, where U is the average velocity in the downstream half channel and L is the width of the downstream half channel, which gives $We = \lambda_1$.

We assume the fully developed Poiseuille flow at the inlet and outlet, the no-slip condition, $u = v = 0$, is applied on the solid boundaries, and $v = 0$ and $\frac{\partial u}{\partial y} = 0$ on the axis of symmetry. The boundary conditions for the stresses along the solid boundaries and inlet are derived from the steady state constitutive equations. At the exit we have Neumann boundary conditions for the stress variables

$$\frac{\partial \tau_{xx}}{\partial x} = \frac{\partial \tau_{yy}}{\partial x} = \frac{\partial \tau_{xy}}{\partial x} = 0.$$

Two different meshes depicted in Fig. 3 were used in the numerical simulations. Mesh1 consists of 5 elements, on each element there are 12 collocation points in the x-direction and 4 collocation points in the y-direction. Mesh2 has 3 elements, there are 18 collocation points in the x-direction and 6 collocation points in the y-direction on each element. We can see that the meshes created by the spectral element method are non-uniform, being refined near the re-entrant corner singularity.



(a) (b)

Fig. 3. Meshes for the four-to-one planar contraction problem: (a) Mesh1; (b) Mesh2.

The numerical stability has been tested for the Newtonian flow ($\lambda_1 = 0$) based on a $(\mathbf{u}, p, \boldsymbol{\tau}, \mathbf{d})$ formulation and numerical results agree well with the corresponding calculation by the velocity-pressure formulation. Fig. 4 shows contours of the stream function and the velocity profiles.

Now we consider the calculations in the viscoelastic case. The results on all the meshes have been computed with $\Delta t = 0.001$ and $Re = 1$. The length of the salient corner vortex L_1, the width of the salient corner vortex L_2 and the maximum value of the stream function φ_{max} are shown in Table 1 for We from 0.1 to 1.2. We found that when We increases from 0 to 0.6, the length of the corner vortex, L_1, is constant, while the width of the corner vortex, L_2, is increased. But when We increases from 0.7 to 1.2, L_1 decreases slightly, and L_2 remains constant. The size of corner vortex compares well quantitatively with the results of [9,11]. Contour plots of vorticity for $We = 0.1, 0.4, 0.8, 1.0$ in Mesh1 are shown in Fig. 5. These vorticity plots show that our numerical results are in good agreement with those obtained by [11].

The streamlines are plotted in Fig. 6 for $We = 0.1, 0.4, 0.8, 1.0$. In Fig. 7 the values of total stress components τ_{xy}, τ_{xx} and τ_{yy} along the line $y = -1$ are given for $We = 0.1, 0.4, 0.8, 1.0$. The maximum values of τ_{xy} and τ_{yy} at the corner are slightly increased when the value of We is increased. A huge increase occurs in the value of τ_{xx} from approximately 4.5 when $We = 0.1$, to approximately 49 when $We = 1.0$.

All accurate results have been presented up to $We = 1.2$. Since for high We number, it becomes more difficult to obtain fully developed velocity and stress fields, further work needs to be done in this area.

Table 1. Values of L_1, L_2 and φ_{max} for various We number with Mesh1.

We	L_1	L_2	φ_{max}
0.1	1.3093	1.086	1.0010672
0.2	1.3093	1.108	1.0010955
0.3	1.3093	1.129	1.0011469
0.4	1.3093	1.140	1.0011860
0.5	1.3093	1.151	1.0012160
0.6	1.3093	1.151	1.0012207
0.7	1.229	1.162	1.0012093
0.8	1.229	1.173	1.0012238
0.9	1.229	1.173	1.0012011
1.0	1.229	1.173	1.0011356
1.1	1.176	1.173	1.0010624
1.2	1.176	1.173	1.0009739

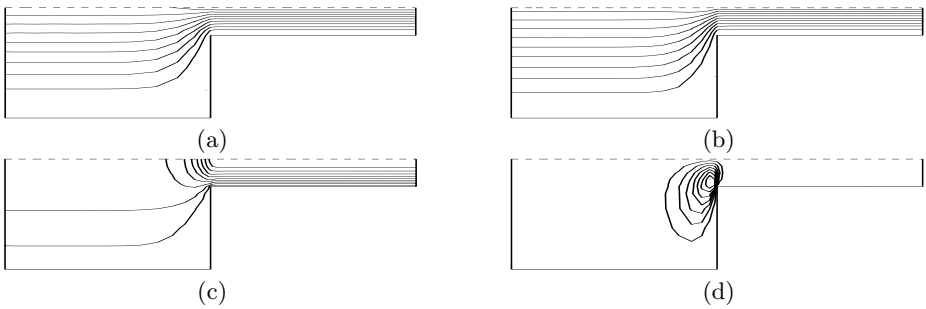


Fig. 4. Numerical stability for the Newtonian flow: (a) streamlines with Mesh1; (b) streamlines with Mesh2; (c) velocity profile in the x-direction with Mesh2; (d) velocity profile in the y-direction with Mesh2.

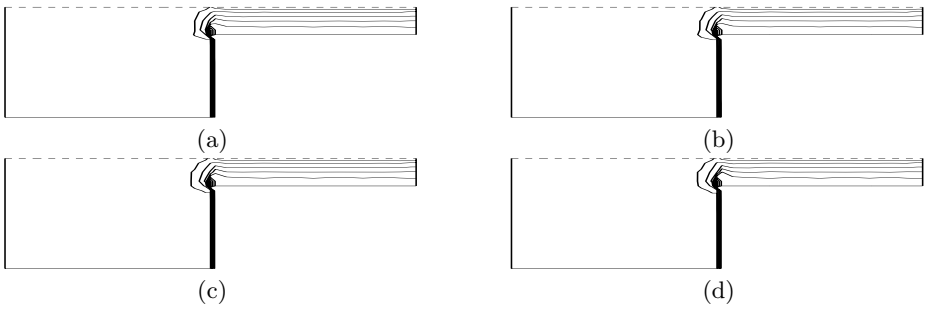


Fig. 5. Vorticity plots for increasing values of We for the viscoelastic flow problem with Mesh1: (a) $We = 0.1$; (b) $We = 0.4$; (c) $We = 0.8$; (d) $We = 1.0$.

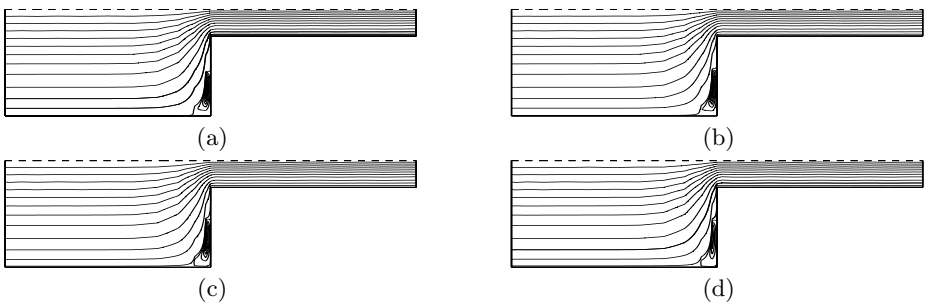


Fig. 6. Streamlines for increasing values of We for the viscoelastic flow problem with Mesh1: (a) $We = 0.1$; (b) $We = 0.4$; (c) $We = 0.8$; (d) $We = 1.0$.

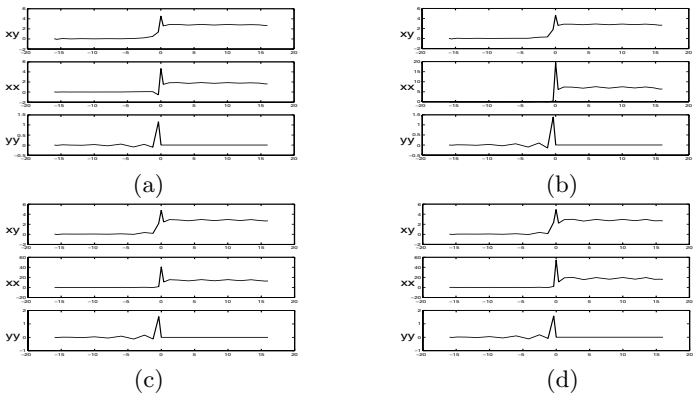


Fig. 7. The values of τ_{xy}, τ_{xx} and τ_{yy} along the line $y = -1$ for increasing values of We for the viscoelastic flow problem with Mesh1: (a) $We = 0.1$; (b) $We = 0.4$; (c) $We = 0.8$; (d) $We = 1.0$.

Acknowledgements
Sha Meng acknowledges the financial support of Ph.D studentship of De Montfort University.

References

1. D. V. Boger. Viscoelastic flows through contractions. *Ann. Rev. Fluid Mech.*, 19:157 182, 1987.
2. F. Brezza. On the existence: uniqueness and approximation of saddle-point problems arising from Lagrange multipliers. *RAIRO Anal. Numer.*, 8 R2:129 151, 1974.
3. M. Fortin and A. Fortin. A new approach for the FEM simulation of viscoelastic flows. *J. Non-Newtonian Fluid Mech.*, 32:295 310, 1989.
4. T. J. R. Hughes. Recent progress in the development and understanding of SUPG methods with special reference to the compressible Euler and Navier-Stokes equations. *Int. J. Num. Methods Fluids*, 7:1261 1275, 1987.
5. R. C. King, M. R. Apelian, R. C. Armstrong, and R. A. Brown. Numerical stable nite element techniques for viscoelastic calculations in smooth and singular geometries. *J. Non-Newtonian Fluid Mech.*, 29:147 216, 1988.
6. Y. Maday and A. T. Patera. Spectral element methods for the incompressible Navier-Stokes equations. *in State of the Art Surveys in Computational Mechanics*, pages 71 143, 1989.
7. J. M. Marchal and M. J. Crochet. Hermitian nite elements for calculating viscoelastic flow. *J. Non-Newtonian Fluid Mech.*, 20:187 207, 1986.
8. J. M. Marchal and M. J. Crochet. A new mixed nite element for calculating viscoelastic flow. *J. Non-Newtonian Fluid Mech.*, 26:77 115, 1987.
9. H. Matallah, P. Townsend, and M. F. Webster. Recovery and stress-splitting schemes for viscoelastic flows. *J. Non-Newtonian Fluid Mech.*, 75:139 166, 1998.
10. D. Ralagopalan, R. C. Armstrong, and R. A. Brown. Finite element methods for calculation of steady, viscoelatic flow using constitutive equations with a Newtonian viscosity. *J. Non-Newtonian Fluid Mech.*, 36:159 192, 1990.
11. T. Sato and S. M. Richardson. Explicit numerical simulation of time-dependent viscoelastic flow problem by a nite element/ nite volume method. *J.Non-Newtonian Fluid Mech.*, 51:249 275, 1994.

SSE Based Parallel Solution for Power Systems Network Equations

Y.F. Fung[1], M. Fikret Ercan[2] ,T.K. Ho[1], and W.L. Cheung[1]

[1] Dept. of Electrical Eng., The Hong Kong Polytechnic University, Hong Kong SAR
{eeyffung, eetkho, eewlcheung}@polyu.edu.hk
[2] School of Electrical and Electronic Eng., Singapore Polytechnic, Singapore
mfercan@sp.edu.sg

Abstract. Streaming SIMD Extensions (SSE) is a unique feature embedded in the Pentium III class of microprocessors. By fully exploiting SSE, parallel algorithms can be implemented on a standard personal computer and a theoretical speedup of four can be achieved. In this paper, we demonstrate the implementation of a parallel LU matrix decomposition algorithm for solving power systems network equations with SSE and discuss advantages and disadvantages of this approach.

1 Introduction

Personal Computer (PC) or workstation is currently the most popular computing system for solving various engineering problems. A major reason is the cost-effectiveness of a PC. With the advanced integrated circuit manufacturing processes, the computing power that can be delivered by a microprocessor is increasing. Currently, processor with a working frequency of 1GHz is available. The computing performance of a microprocessor is primarily dictated by two factors, namely the operating frequency (or clock rate), and the internal architecture.

The Streaming SIMD Extensions (SSE) is a special feature available in the Intel Pentium III class of microprocessors. As its name implies, the SSE enables the execution of SIMD (Single Instruction Multiple Data) operations inside the processor and therefore, the overall performance of an algorithm can be improved significantly.

V.N. Alexandrov et al. (Eds.): ICCS 2001, LNCS 2073, pp. 862-873, 2001.
© Springer-Verlag Berlin Heidelberg 2001

The power network problem is computationally intensive and in order to reduce the computation time many researchers have proposed solutions [1,2] based on parallel hardware systems. However, most of those hardware platforms are expensive and may not be available to most researchers. On the other hand, the cost of a PC is low and therefore, an improved solution to the power network problem utilizing SSE will benefit to research in this area. In the next section, details of the SSE mechanism will be described, followed by a discussion on the problem of power systems network solution. The parallel algorithm using SSE and its performance will be discussed consecutively.

2 SSE Mechanism

The SSE can be considered as an extension of the MMX technology implemented by the Intel Pentium processors [3]. It provides a set of 8 64-bit wide MMX registers and 57 instructions for manipulating packed data stored in the registers.

2.1 Register and Data Storage

The major difference between SSE and MMX is in the data-type that can be operated upon in parallel. In MMX, special MMX registers are provided to hold different types of data, however, it is limited to character, or integer values.

On the other hand, the SSE registers are 128-bit wide and they can store floating-point values, as well as integers. There are eight SSE registers, each of which can be directly addressed using the register names [4]. Utilization of the registers is straightforward with a suitable programming tool. In the case of integers, eight 16-bit integers can be stored and processed in parallel. Similarly, four 32-bit floating-point values can be manipulated. Therefore, when two vectors of four floating-point values have been loaded into two SSE registers, as shown in Fig. 1, SIMD operations, such as add, multiply, etc., can be applied to the two vectors in one single operation step. Applications relying heavily on floating-point operations, such as 3D geometry, and video processing can be substantially accelerated [5]. Moreover, the support of floating-point values in the SSE operations has tremendously widened its applications in other problems including the power systems network problem described in this paper.

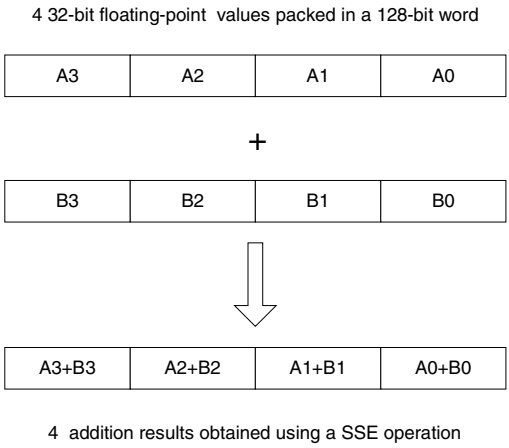


Fig. 1. Parallelism based on SSE operation

2.2 Programming with SSE

Programming with the SSE can be achieved by two different approaches. The SSE operations can be invoked by assembly codes included in a standard C/C++ programs. In following, sample codes showing how to evaluate the value $(1/x)$ using assembly codes are given.

```
          float   x, frcp;
__asm {   movss   xmm1,DWORD PTR x
          movss   xmm2,xmm1
          rcpss   xmm1,xmm1
          movss   xmm3,xmm1
          mulss   xmm1,xmm1
          mulss   xmm2,xmm1
          addss   xmm3,xmm3
          subss   xmm3,xmm2
          movss   DWORD PTR frcp, xmm3}
```

Alternatively, by utilizing the special data type we can develop a C/C++ program without any assembly coding. The new data type designed for the manipulation of the SSE operation is ***F32vec4*** [4]. It represents a 128-bit storage, which can be applied to

store four 32-bit floating-point data. Similarly, there is also the type ***F32vec8***, which is used to store eight 16-bit values. These data types are defined as C++ classes and they can be applied in a C/C++ program directly.

In addition to the new data types, operations are derived to load traditional data, such as floating-point, into the new data structure. As an example, to load (or pack) four floating-point values into a ***F32vec4***, the function _mm_load_ps can be applied. When using _mm_load_ps, it is assumed that the original data is 16-byte aligned (16-byte aligned implies that the memory address of the data is a factor of 16) otherwise the function _mm_loadu_ps should be used instead. Once data are stored into the 128-bit data structure, functions that can manipulate the ***F32vec4*** type data can be called. This will result in parallel processing in two sets of four floating-point values. Source codes demonstrating how to add elements stored in two arrays using the SSE features are depicted as following:

```
Float array1[4];

Float array2[4];

Float result[4];

F32vec4 A1, A2, A3;

A1 = _mm_load_ps(array1);

A2 = _mm_load_ps(array2);

A3 = A1+A2;

_mm_store_ps(result, A3);
```

the variable A1 and A2 can be manipulated just like any standard data type. The function _mm_store_ps is used to convert (or unpack) the data from the ***F32vec4*** type back to floating-points and stored in an array.

3 Power System Network Equations

The power systems network equations usually involve identifying solutions for a set of linear equations in the form of:

$$Ax = b \tag{1}$$

where A is an incidence symmetric sparse matrix of order n, b is a given independent vector and x is an unknown solution vector. As discussed in the introduction, the problem is computationally intensive. In addition, for some applications such as

real-time power systems simulation, solution for equation (1) must be determined in a short time-interval [5], e.g. 10 ms, this also demands a very fast computation.

A common procedure [6] for solving (1) is to factor A into lower and upper triangular matrices L and U such that

$$LUx = b \tag{2}$$

and this then followed by forward/backward substitution of the form

$$Lx' = b \tag{3}$$

and

$$Ux = x' \tag{4}$$

Forward substitution first identifies the intermediate results x' and vector x is determined by backward substitution.

A realistic power system network is comprising of a number of sub networks A_i connected via t_i -lines A_{ic}, as shown in Fig. 2, to a group of busbars known as cut-

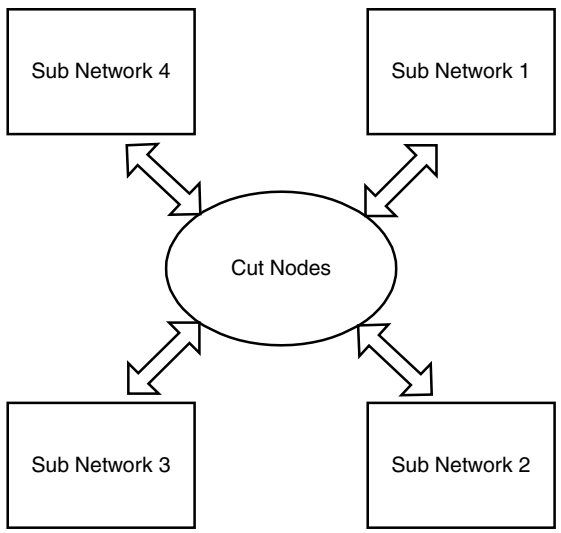


Fig. 2. Block diagram of power systems networks

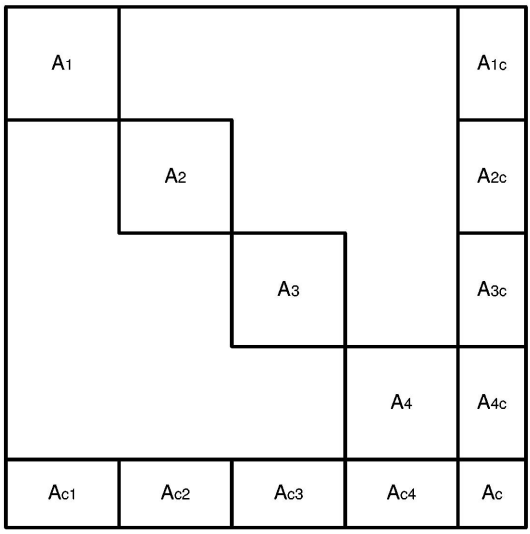


Fig. 3. Bordered block diagonal form for a power network system

nodes A_c [5]. If the network admittance matrix is arranged to follow the sub network configuration, it can be re-arranged into the Bordered Block Diagonal Form (BBDF) as shown in Fig. 3.

The BBDF matrix can now be grouped into sub-matrices, as shown in Fig. 4. Each matrix can be solved by LU decomposition. The solution for the A_c (the cut-node block) is determined by

$$L_c U_c = A_c - \sum_{i=1}^{n} A_{ic} \tag{5}$$

Referring to Fig.4, the sub-matrix is now a dense matrix and therefore, traditional dense matrix algorithm can be applied to determine the L, U triangular matrices. On the other hand, the BBDF, which is a sparse matrix, should be solved by sparse matrix solutions, such as the Choleski method [7].

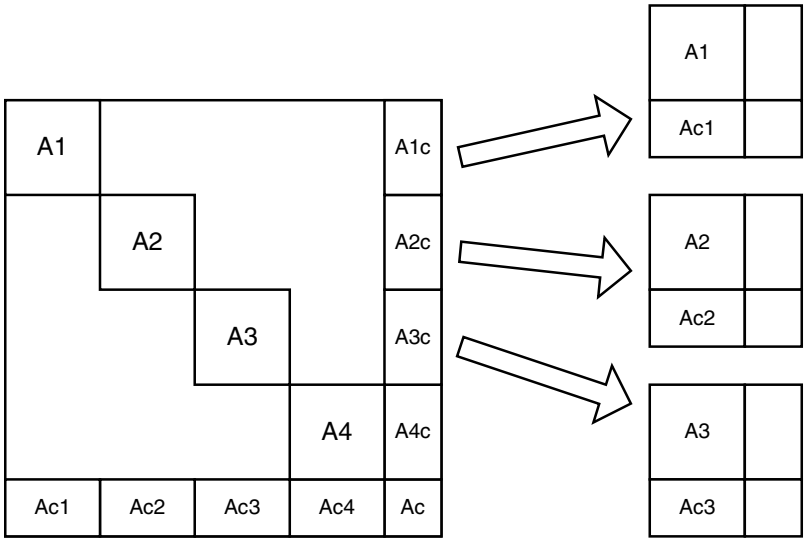


Fig. 4. Partitioning the BBDF matrix into sub-matrices

4 Parallel LU Decomposition Based on SSE

The calculation involved in LU decomposition can be explained by the following equation:

$$
\begin{aligned}
&For \quad k = 0 \ to \ \ n - 2 \\
&Do \\
&For \quad i = k + 1 \ to \ \ n - 1 \\
&Do \\
&For \quad j = k + 1 \ to \ \ n - 1 \\
&a_{i,j} = a_{i,j} - \frac{a_{i,k} \times a_{k,j}}{a_{k,k}}
\end{aligned}
\tag{6}
$$

In the above equation, $a_{i,j}$ represents elements in the A matrix.

According to (6), elements in the matrix A are being processed along the diagonal and on a row-by-row basis. Data stored in a row of the matrix map naturally into the ***F32vec4*** data and therefore, four elements in a row can be evaluated in one single step.

Based on (6), the term $\frac{a_{i,k}}{a_{k,k}}$ is a constant when elements in row i are being processed. It can be, therefore, stored in a ***F32vec4*** value with the command _mm_load_ps1. The command loads a single 32-bit floating-point value, copying it into all four words. The pseudo codes shown in following illustrate the steps performed in order to implement equation (6) using SSE functions.

```
F32vec C, A1, A2;  /* 128-bit values */

Float x;

For (k=0; k<n-2; k++)

For (i=k+1; i<n-1; i++) {x = a_{i,k}/a_{k,k}; _mm_load_ps1(C, x);

for (j=k+1; j<n-1; j+=4){

pack four values from a(k,j) to a(k,j+3) into A1;

pack four value from a(i,j) to a(i,j+3) into A2;

A2 = A2 - (A1*C); Unpack the results from A2 and store
in output matrix

}}
```

In forward substitution, the operations can be represented by:

$$x'_i = b_i - \sum_{j=1}^{i-1} x'_j \bullet L_{i,j} \tag{7}$$

where $[x'_i]$ represents element in the $[x']$ matrix as shown in equation (4); b_i represents elements in the $[b]$ matrix $L_{i,j}$ represents elements in the $[L]$ matrix. SSE operations are applied in the operation $x'_j \bullet L_{i,j}$. Four elements of x'_j and $L_{i,j}$ can be stored in two different ***F32vec4*** data and multiplied at a single operation.

In backward substitution, the operations are represented by

$$x_j = \frac{x'_j - \sum_{n=j+1}^{m} x_n \bullet U_{j,n}}{U_{j,j}} \tag{8}$$

where $U_{i,j}$ are the elements in the Upper matrix $[U]$; m is the size of the vector $[x]$. Similar to forward substitution, the multiplication of $x_n \bullet U_{j,n}$ can be executed by SSE functions with four elements of x_n and $U_{j,n}$ being operated on at the same instead.

5 Experimental Results

In Sections 3 and 4, the processing requirements for the power system network equations and basic features of SSE have been described. In this section, results obtained from the equation $Ax = b$ based on LU decomposition and forward and backward substitutions are given. Processing time obtained for different dimensions of the matrix A are given in Table 1. Three different cases are being compared, namely, (1) conventional approach that is without using SSE, (2) solution obtained by SSE, (3) solution obtained by SSE (but without the 16-bit alignment.) The speedup ratios, by taking the processing time of the traditional approach as reference, are illustrated in Fig. 5.

Table 1. Processing time for solution of $Ax = b$ in (ms)

	Size of Matrix A		
	100	200	400
Traditional	9	58.83	525,2
SSE	6	40	417
SSE with alignment	5	36.83	367.8

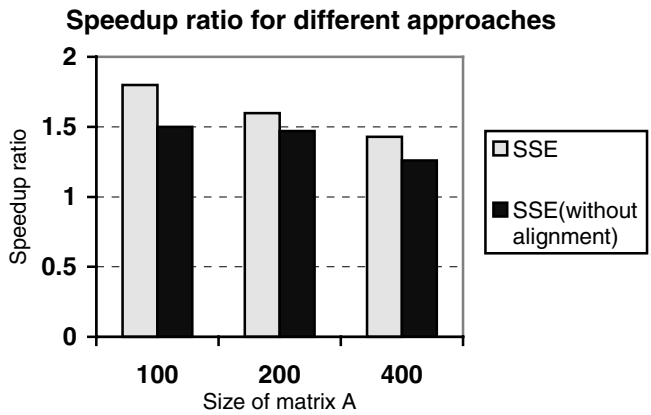


Fig. 5. Speedup ratio for three different approaches

Referring to Fig. 5, the speedup ratio obtained in the case of using SSE is slightly better than non-aligned case. For a better performance, data should be 16-byte aligned form as described in Section 2.2. In order to maintain the 16-byte alignment, the size of the matrix must be a multiple of 4. If this is not the case, then extra rows and columns must be added into the matrix to get around the alignment problem.

The best result is obtained for a relatively smaller matrix of 100x100 where the speedup rate is about 1.8. The performance of the SSE algorithms is affected by the overhead due to additional steps required to convert data from standard floating-point values to 128-bit ***F32vec4*** data and vice-versa. Referring to pseudo code given in section 4, three packing/unpacking functions are carried out when elements in a row are being processed. Assuming that it takes the same time (t_p) to execute a packing, or unpacking function, then the overhead will be $3t_p$. And the time required to operate on four elements becomes $3t_p + t_m$, where t_m is the processing time for one multiplication, one subtraction, and one store operation with SSE. If we define $(3t_p + t_m)$ as t_{sse}, then the total processing time becomes $t_{sse} \times$(total number of operations). In the case of SSE, the number of operations can be appropriated by

$$\frac{1}{4}\sum_{n=2}^{N-1}(n-1)(n) \tag{9}$$

where N is the size of the vector x given in equation (1).

In the case of traditional algorithm, the total number of operations is

$$\sum_{n=2}^{N-1}(n-1)^2 \tag{10}$$

and the processing time per operation is t'_m, which is the time taken to perform the multiplication, subtraction and store with the standard programming approach. And we can assume that the processing time for t_m and t'_m are the same.

The equation (9) does not include processing in the forward and backward substitution. The forward and backward substitution only account for a very small portion (about 1%) of the total processing, therefore it is neglected in the current model.

According to equations (9) and (10), the speedup ratios obtained for different sizes of the matrix A can be approximated by a constant, provided that the values of $3t_p$ and t_m are known. The values of $3t_p$ and t_m are being determined empirically

and our result indicates that $3*t_p \approx t_m$. The speedup ratios obtained by our model is close to our experimental results and therefore, we can gauge the performance of the SSE algorithm with different sizes of matrix A based on our model.

6 Conclusions

In this paper, we have presented the basic operations involved in utilizing the SSE features of the Pentium III processors. In order to examine the effectiveness of SSE, the power systems network equations problem was solved by applying SSE functions. According to our results, a speedup ratio around 1.5 can be easily obtained. The results are satisfactory and only minor modifications of the original program are needed in order to utilize the SSE features. Most importantly, additional hardware is not required for this performance enhancement. SSE is a valuable tool for improving the performance of computation intensive problems, and the power systems network equations problem is one of the ideal applications.

In this paper, we have only considered the DC power systems, where computations involve only real numbers. Currently, we are investigating AC systems that require processing of complex numbers. Moreover, the applications of SSE in a dual-CPU system will also be studied. These studies will lead to cost-effective high performance computation of power system applications.

References

1. Taoka, H., Iyoda, I., and Noguchi, H.: Real-time Digital Simulator for Power System Analysis on a Hybercube Computer. IEEE Trans. On Power Systems. 7 (1992) 1-10
2. Guo Y., Lee H.C., Wang X., and Ooi B.: A Multiprocessor Digital Signal Processing System for Real-time Power Converter Applications, IEEE Trans. On Power Systems, 7 (1992) 805-811
3. The Complete Guide to MMX Technology, Intel Corporation, McGraw-Hill (1997)
4. Conte G., Tommesani S., Zanichelli F.: The Long and Winding Road to High-performance Image Processing with MMX/SSE, IEEE Int'l Workshop for Computer Architectures for Machine Perception (2000), 302-310.
5. Wang K.C.P., and Zhang X.: Experimentation with a Host-based Parallel Algorithm for Image Processing, Proc. 2nd Int'l conf on Traffic and Transportation Studies (2000) 736-742.
6. Intel C/C++ Compiler Class Libraries for SIMD Operations User's Guide (2000)
7. Chan K.W. and Snider, L.A.: Development of a Hybrid Real-time Fully Digital Simulator for the Study and Control of Large Power Systems, Proc. of APSCOM 2000, Hong Kong, (2000) 527-531

8. Wu J.Q., and Bose A.: Parallel Solution of Large Sparse Matrix Equations and Parallel Power Flow, IEEE Trans. On Power Systems, 10 (1995) 1343-1349
9. Jess J.A., and Kees, G.H.: A Data Structure for Parallel LU Decomposition, IEEE Trans., C-31 (1992) 231-239

Implementation of Symmetric Nonstationary Phase-Shift Wavefield Extrapolator on an Alpha Cluster

Yanpeng Mi and Gary F. Margrave

CREWES Project, Geology and Geophysics Department, The University of Calgary, 2500 University Drive, Calgary, Alberta T2N 1N4, Canada
mi@geo.ucalgary.ca, gary@geo.ucalgary.ca

Highly accurate seismic depth imaging (migration) is required by the oil/gas exploration industry in regions of complex geology. Fourier-domain depth-imaging techniques provide the needed accuracy although the cost of computation can be large when the lateral velocity variation is rapid. Parallel computers provide an efficient solution to overcome the computational barrier of Fourier-domain imaging techniques. In this paper, a new Fourier-domain imaging technique, based on nonstationary filtering and wavefield extrapolation theories, is described. The amount of computation required for an example 2D synthetic seismic data set is analyzed. The parallel version of the algorithm is implemented using C++ and Fortran 90 on an Alpha cluster workstation. The Message Passing Interface (MPI) library was used for data distribution and collection. The algorithm, found to be accurate but slow on serial machines, can achieve a speed acceptable for most industrial applications.

Introduction

Seismic depth imaging repositions the scattered energy from the subsurface to its correct spatial location according to a velocity model. Produced by a true-amplitude depth imaging technique, a subsurface image also carries amplitude information that is directly proportional to reflection coefficients.

Kirchhoff depth-imaging techniques are very common in the exploration industry due to robustness and high efficiency. Theoretically the method is based on Green's theorem and it is implemented by summation along iso-time surfaces computed by raytracing. Difficulties arise when it is used in regions of complex geology. Fourier-domain imaging algorithms, especially for complex media, are often slower than Kirchhoff-type techniques. However, they provide a higher accuracy solution by allowing the energy to propagate in all possible directions instead of only the Snell's law paths. Fourier-domain techniques can be made efficient by making reasonable approximations and operating in efficient data domains.

V.N. Alexandrov et al. (Eds.): ICCS 2001, LNCS 2073, pp. 874–883, 2001.
© Springer-Verlag Berlin Heidelberg 2001

Fourier domain wavefield extrapolation

The wavefield at depth z can be obtained by extrapolation the wavefield at depth 0, given knowledge of the velocity field between 0 and z. Consider a mono-frequency wavefield at depth 0, $\varphi(k_x, z=0, \omega)$, which has been 2D forward Fourier transformed to frequency-wavenumber (ω,k_x) domain, the wavefield at depth z can be written as (Gazdag, 1978),

$$\varphi(k_x, z, \omega) = \varphi(k_x, 0, \omega)\alpha(k_x, z, \omega), \tag{1}$$

where $\alpha(k_x, z, \omega)$ is the (ω,k_x) domain wavefield extrapolator operating from 0 to z.

For homogeneous and isotropic media, the wavefield extrapolator is known analytically and the extrapolation step size is unlimited. For velocity v, α can be written as

$$\alpha(k_x, z, \omega) = e^{\pm iz\sqrt{\frac{\omega^2}{v^2} - k_x^2}}, \tag{2}$$

where the plus and minus signs denote upward and downward extrapolation when z axis points downward. For heterogeneous media, the extrapolation operator is known only approximately and the step size is limited. The extrapolation is done recursively with small steps, so that local wave propagation can be approximated by homogeneous and isotropic propagation theory.

There are several ways to compute an approximate wavefield extrapolator for heterogeneous media. Typical algorithms are phase-shift-plus-interpolation (PSPI) originally developed by Gazdag and Sguazzero (1984) and the split-step Fourier algorithm originally developed by Stoffa et al. (1990), which is also called the phase-screen algorithm by Wu and Huang (1992). The PSPI algorithm uses several constant velocities to produce reference extrapolated wavefields. The wavefield at any spatial location is computed by linear frequency-space (ω,x) domain interpolation between the reference wavefields according to the velocity function $v(x)$. The split-step Fourier algorithm assumes the lateral velocity variation is small so that perturbation theory applies. It first applies a homogeneous wavefield extrapolation with a constant reference velocity and then performs a (ω,x) domain slowness perturbation correction to accommodate lateral velocity variation. There are many further developments of the above algorithms in order to deal with extreme lateral velocity variation and achieve high efficiency (Jin and Wu, 1998; Popovici, 1996; etc.) These algorithms have also been implemented on parallel computers, for example, Tanis and Stoffa (1997).

Nonstationary Fourier wavefield extrapolators

Using a complete set of reference velocities, the PSPI algorithm becomes an integral over horizontal wave number k_x, which performs wavefield extrapolation simultaneously with an inverse Fourier transform (Margrave and Ferguson 1999a)

$$\psi(\mathrm{x},z,\omega)=\frac{1}{(2\pi)^2}\int_{-\infty}^{+\infty}\alpha(\mathrm{x},\mathrm{k}_x,z,\omega)\varphi(\mathrm{k}_x,0,\omega)\exp(-i\mathrm{k}_x x)d\mathrm{k}_x\,, \tag{3}$$

where $\alpha(\mathrm{x},\mathrm{k}_x,z,\omega)$ is the nonstationary wavefield extrapolator

$$\alpha(\mathrm{x},\mathrm{k}_x,z,\omega)=e^{\pm iz\sqrt{\frac{\omega^2}{v^2(x)}-k_x^2}}\,. \tag{4}$$

$\psi(\mathrm{x},z,\omega)$ is the (ω,x) domain expression of the wavefield at depth z and $\varphi(\mathrm{k}_x,0,\omega)$ is the (ω,k_x) domain expression of the wavefield at depth 0. The meanings of plus and minus signs are defined as before. Though there is no interpolation in equation (3), the expression PSPI is used for equation (3) because it is a limiting form of Gazdag's PSPI.

Equation (3) has a complementary form, called nonstationary phase-shift (NSPS), which performs wavefield extrapolation simultaneously with a forward Fourier transform (Margrave and Ferguson, 1999a)

$$\varphi(\mathrm{k}_x,z,\omega)=\int_{-\infty}^{+\infty}\alpha(\mathrm{x},\mathrm{k}_x,z,\omega)\psi(\mathrm{x},0,\omega)\exp(i\mathrm{k}_x x)d\mathrm{x}\,, \tag{5}$$

where $\alpha(\mathrm{x},\mathrm{k}_x,z,\omega)$ is expressed by equation (4).

Equation (3) can be approximately computed by matrix-vector multiplication, which can be written as

$$\underline{\psi}_z=\underline{\underline{A}}\,\underline{\varphi}_0\,, \tag{6}$$

where $\underline{\varphi}_0$ and $\underline{\psi}_z$ are column vectors representing a mono-frequency wavefield in (ω,k_x) domain at depth 0 and the extrapolated wavefield in (ω,x) domain at depth z, respectively. Matrix $\boldsymbol{A}$ is the combination of the wavefield extrapolator and the simultaneous inverse Fourier-transform kernel

$$\underline{\underline{A}} = \begin{pmatrix} e^{i\left(\pm z\sqrt{\omega^2/v_1^2 - k_{x1}^2} - x_1 k_{x1}\right)} & \cdots & e^{i\left(\pm z\sqrt{\omega^2/v_1^2 - k_{xn}^2} - x_1 k_{xn}\right)} \\ \vdots & \vdots & \vdots \\ e^{i\left(\pm z\sqrt{\omega^2/v_n^2 - k_{x1}^2} - x_n k_{x1}\right)} & \cdots & e^{i\left(\pm z\sqrt{\omega^2/v_n^2 - k_{xn}^2} - x_n k_{xn}\right)} \end{pmatrix}. \quad (7)$$

Similarly, equation (5) can be approximately written in a matrix form

$$\underline{\varphi}_z = \underline{\underline{B}}\,\underline{\psi}_0, \quad (8)$$

where $\underline{\psi}_0$ and $\underline{\varphi}_z$ are column vectors representing a mono-frequency wavefield in (ω,x) domain at depth 0 and the extrapolated wavefield in (ω,k_x) domain at depth z, respectively. Matrix **B** is the combination of the wavefield extrapolator and the simultaneous forward Fourier-transform kernel

$$\underline{\underline{B}} = \begin{pmatrix} e^{i\left(\pm z\sqrt{\omega^2/v_1^2 - k_{x1}^2} + x_1 k_{x1}\right)} & \cdots & e^{i\left(\pm z\sqrt{\omega^2/v_n^2 - k_{x1}^2} + x_n k_{x1}\right)} \\ \vdots & \vdots & \vdots \\ e^{i\left(\pm z\sqrt{\omega^2/v_1^2 - k_{xn}^2} + x_1 k_{xn}\right)} & \cdots & e^{i\left(\pm z\sqrt{\omega^2/v_n^2 - k_{xn}^2} + x_n k_{xn}\right)} \end{pmatrix}. \quad (9)$$

Margrave and Ferguson (1999b) showed that the NSPS and PSPI can be naturally combined into a symmetric wavefield extrapolator (SNPS) by first performing NSPS for the upper half z and PSPI for the lower half z within a single step. A Taylor series derivation of PSPI and NSPS and related error analysis showed that the first-order errors of PSPI and NSPS oppose one another, so that SNPS has a smaller error and is more stable than either PSPI or NSPS alone (Margrave and Ferguson, 2000).

Prestack depth imaging of a reflector requires that the wavefield recorded on the Earth's surface be extrapolated backward and a model of the seismic source be extrapolated forward. At any particular depth, reflectivity is estimated as the ratio of the backward extrapolated recorded data (reflected wave) to the forward extrapolated source model (incident wave). To avoid a frequency dependent estimate, an average over frequency is used

$$\tilde{r}(x,z)=\frac{1}{[\omega_{max}-\omega_{min}]}\int_{\omega_{min}}^{\omega_{max}}\left\{\frac{\psi^{*}_{R}(x,z,\omega)}{\psi^{*}_{S}(x,z,\omega)}+\frac{\psi_{R}(x,z,\omega)}{\psi_{S}(x,z,\omega)}\right\}d\omega \qquad (10)$$

where ψ_S and ψ_R are the incident and reflected wavefields immediately above the reflector at depth z, and * denotes complex conjugation.

Computational considerations for shot gather depth imaging

Accurate depth imaging with SNPS requires that equations (3) and (5) are computed at every depth step. To study the amount of computation, a synthetic seismic data set called Marmousi (Bourgeois et al., 1991) was used. The original data set was computed by finite difference modeling. It simulates marine acquisition and has 240 shot gathers. Each shot gather has 96 traces whose offsets vary from 200 m to 2600 m. Each trace has 512 samples and the sampling rate is 4 ms. Depth imaging was done with 752 downward steps of 4 m each. The shot gathers were padded to 256-trace split-spread geometry to accommodate the migrated energy.

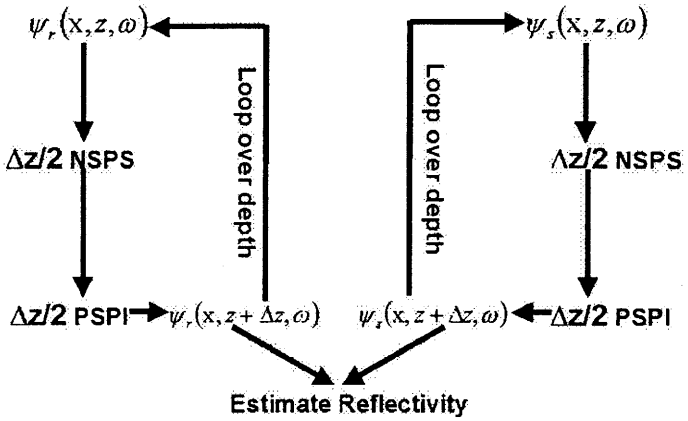


Fig. 1. Computational steps of the SNPS depth imaging algorithm.

The computing steps of SNPS are shown in Figure 1. Assume there are N traces in a shot gather and the number of frequencies needs to be migrated, N_ω, is of the same order of N, each step of SNPS is an N^3 algorithm since each frequency needs to be computed by a matrix-vector multiplication, which is an N^2 algorithm. Assume the number of depth steps is N_z and the number of shots is N_s, the total computation required for an imaging project is of the order of $N_s N_z N^3$.

It took about half an hour to migrate a single shot gather of the Marmousi data on an single-CPU, 128 Mb memory Alpha XP 1000 workstation and the full migration of the 240 shot gathers requires 120 hours of computing time. This speed is not acceptible for industrial use though it produced a better depth

image than commonly used algorithm in the industry. A practical approach is to compute the local approximants of the integral operators by spatial windowing. The result from each window is then collected by either a inverse spatial Fourier transform (for PSPI) or a forward spatial Fourier transform (for NSPS). Consider extrapolation from $z=0$ to depth z, the PSPI and NSPS integrals can be approximated by

$$\Psi(x,z,\omega)=\sum_j \beta_j(x,\omega)\Omega_j \frac{1}{2\pi}\int_{-\infty}^{\infty}\alpha_j(k_x,z,\omega)\varphi(k_x,0,\omega)\exp(-ik_x x)dk_x \,, \quad (13)$$

which corresponds to a filtering-then-windowing process, and

$$\varphi(k_x,z,\omega)=\sum_j \alpha_j(k_x,z,\omega)\int_{-\infty}^{\infty}\Psi(x,0,\omega)\Omega_j\beta_j(x,\omega)\exp(ik_x x)dx \,, \quad (14)$$

which corresponds to a windowing-then-filtering process. The windowing function Ω_j is defined as

$$\Omega_j(x)=\begin{cases}1, v(x)=v_j\\ 0, otherwise\end{cases}. \quad (15)$$

$\alpha_j(k_x,z,\omega)$ is defined as

$$\alpha_j(k_x,z,\omega)=\exp\left(\pm iz\sqrt{\left(\frac{\omega}{v_j}\right)^2-k_x^2}\right), \quad (16)$$

and $\beta_j(x,\omega)$ is (ω,x) domain vertical traveltime correction,

$$\beta_j(\omega,x)=\exp\left(\pm i\omega z\left(v^{-1}(x)-v_j^{-1}\right)\right). \quad (17)$$

A recursive symmetric windowed wavefield extrapolator can then be formed by combining equations (13) and (14). Computation of equation (13) is much faster than that of the SNPS integral operator since for each frequency, the N^2 order matrix-vector multiplication is replaced by a Fourier transform of order NlogN. Assume the number of depth steps is N_z and the number of shots is N_s, and if only one reference velocity is used, the total computation required for an imaging project becomes the order of $N_sN_zN^2\log N$. For extremely complex media, more than one reference velocity should be used, so that the total computation becomes an algorithm of order $N_sN_zN_vN^2\log N$, where N_v is the number of reference velocities.

It took about 4 minutes to migrate a shot gather on the Alpha XP 1000 workstation, on which the integral algorithm is tested. However, the image quality is slightly inferior to that produced by the SNPS integral extrapolators.

Parallel implementation on MACI Alpha Cluster

Both the integral SNPS extrapolation algorithm and its approximations were implemented on the Multimedia Advanced Computational Infrastructure (MACI) Alpha Cluster at the University of Calgary. MACI is a collaborative project of the Universities of Alberta, Calgary, Lethbridge and Manitoba that provides high performance computing resources to Alberta researchers and educators. The MACI Alpha Cluster at the University of Calgary is part of the high-performance computing facility of the MACI project. It consists of 128 Compaq Alpha workstations and each single user can use up to 16 CPUs for a single computing task. A general network configuration of the Alpha Cluster is shown in figure 2.

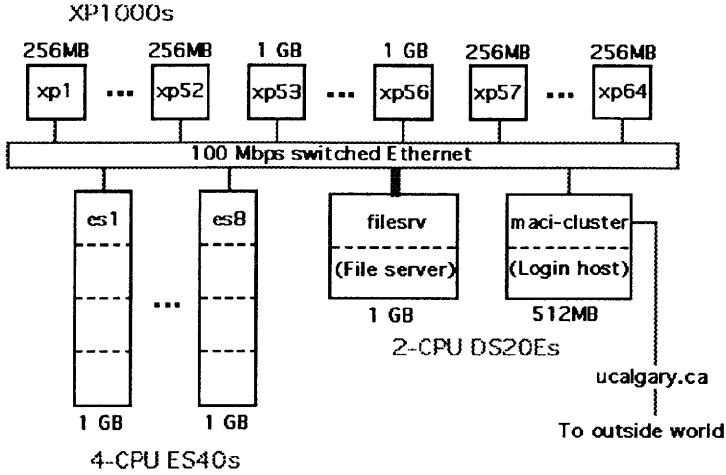
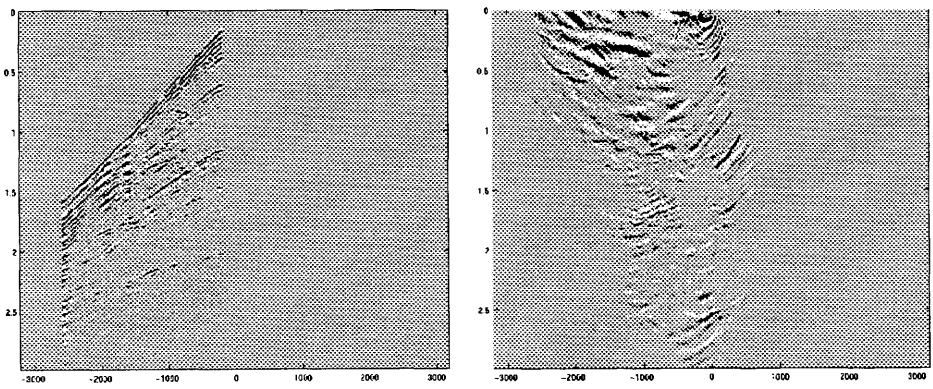


Fig. 2. The general network configuration of the Alpha Cluster at the University of Calgary.

The Message Passing Interface (MPICH1.2) was used for parallel implementation. Each computing node was assigned a single shot gather migration task. The master C++ main program performs data I/O and parallel data distribution/collection. The shot gather imaging kernel function was written in FORTRAN 90 for high performance. It took about 1 hour to migrate the 240 shot gathers on 16 XP1000 workstations. Figure 3 shows shot gather 120 that lies in the middle of the model before and after migration. Each shot gather migration produces a local image of the subsurface and overlapping all the migrated shot gathers produces a complete subsurface image. Figure 4 shows the 0-20 Hz band-limited reflectivity of the Marmousi model and figure 5 shows the subsurface image generated by the integral SNPS extrapolator and that generated by its approximation. Note that the approximate algorithm produced an image only slightly inferior to that generated by the integral algorithm. The three fault planes in the middle of the model and the target flat spot at depth 2550 m from x=6000 m to x=7500 m are imaged well.

Conclusions and discussion

The integral SNPS extrapolator produces a very accurate subsurface image. The approximate implementation has higher speed with very little sacrifice of image quality. The speed is directly proportional to the number of reference velocities. Direct computation of equation (3) and (5) can be made faster by utilizing the symmetric properties of the extrapolator and the forward/inverse Fourier transform kernels. Faster while less accurate numerical algorithms to compute the square root and exponential function can also be used to improve the computing performance.



(a) (b)

Fig. 3. Shot gather 120 before (a) and after migration (b).

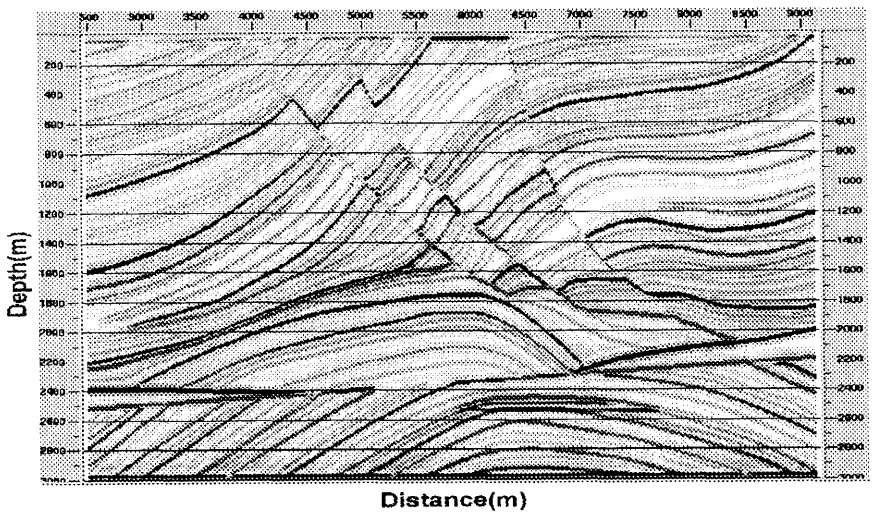
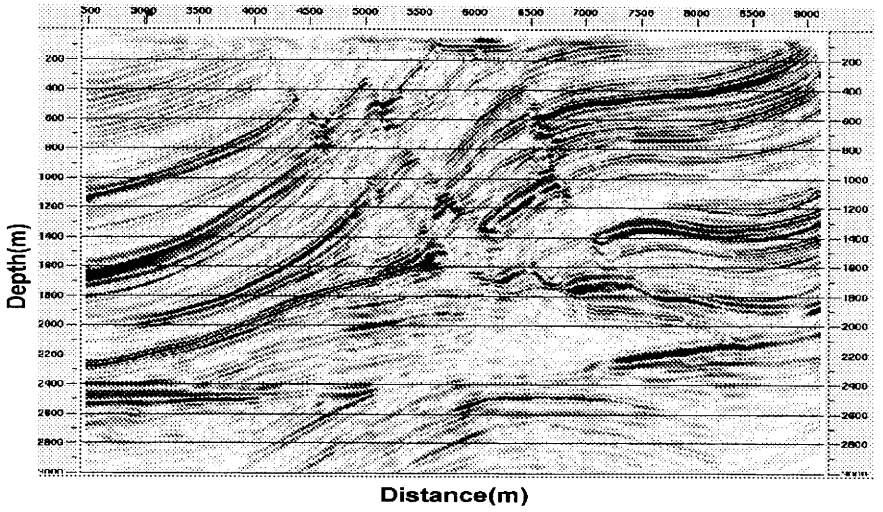
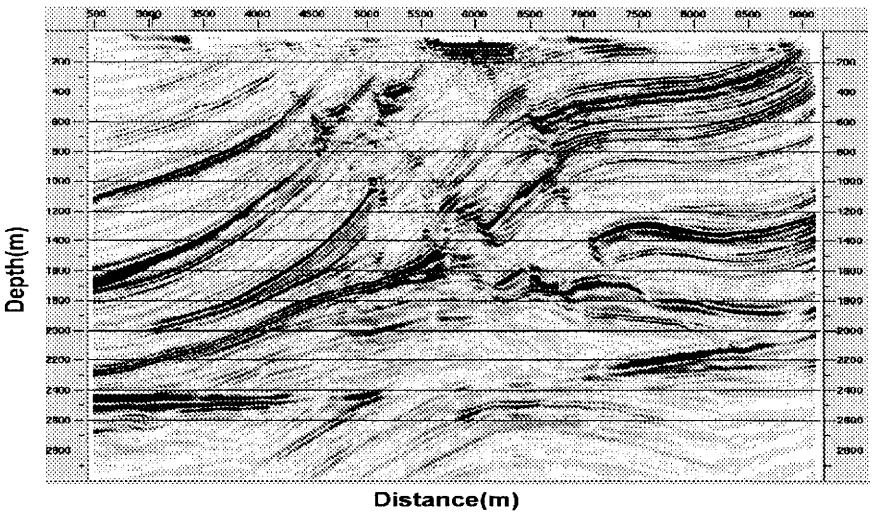


Fig. 4. 0-20 Hz band-limited reflectivity of the Marmousi model.



(a)



(b)

Fig. 5. (a) subsurface image produced by integral SNPS extrapolator; (b) subsurface image produced by the approximate algorithm. Only frequency content of 5-40 Hz was migrated.

References

Bourgeois, A., Bourget, M., Lailly, P., Poulet, M., Ricarte, P., and Versteeg, R., 1991, Marmousi, model and data: in The Marmousi experience, EAGE, 5 - 9.

Ferguson, R. and Margrave, G., 1999, A practical implementation of depth migration by nonstationary phase shift: Annual Meeting Abstracts, Society Of Exploration Geophysicists, 1370-1373.

Gazdag, J., 1978, Wave equation migration with the phase-shift method, Geophysics, **43**,1342-1351.

Gazdag, J. and Sguazzero, P., 1984, Migration of seismic data by phase shift plus interpolation, Geophysics, **49**, 124-131.

Jin, S. and Wu, R., 1998, Depth migration using the windowed generalized screen propagators: Annual Meeting Abstracts, Society Of Exploration Geophysicists, 1843-1846.

Margrave, G. F. and Ferguson, R. J., 1999a, Wavefield extrapolation by nonstationary phase shift: Geophysics, **64**, no. 04, 1067-1078.

Margrave, G. and Ferguson, R., 1999b, An explicit, symmetric wavefield extrapolator for depth migration: Annual Meeting Abstracts, Society Of Exploration Geophysicists, 1461-1464.

Margrave, G. and Ferguson, R., 2000, Taylor series derivation of nonstationary wavefield extrapolators: Annual Meeting Abstracts, Society Of Exploration Geophysicists, 834-837

Popovici, A. M., 1996, Prestack migration by split-step DSR: Geophysics, **61**, no. 05, 1412-1416.

Stoffa, P. L., Fokkema, J. T., de Luna Freire, R. M. and Kessinger, W. P., 1990, Split-step Fourier migration: Geophysics, **55**, 410-421.

Tanis, M. C. and Stoffa, P. L., 1997, Parallel implementation of 3-D split-step Fourier depth migration algorithm on T3E: Annual Meeting Abstracts, Society Of Exploration Geophysicists, 1433-1436.

Wu, R. and Huang, L., 1992, Scattered field calculation in heterogeneous media using phase-screen propagation: Annual Meeting Abstracts, Society Of Exploration Geophysicists, 1289-1292.

Generalized High-Level Synthesis of Wavelet-Based Digital Systems via Nonlinear I/O Data Space Transformations

Dongming Peng and Mi Lu

Electrical Engineering Department, Texas A&M University, College Station, TX77843, USA

1 Introduction

In this paper, we systematically present the high-level architectural synthesis for general wavelet-based algorithms via a model of I/O data space and the nonlinear transformations of the I/O data space. The parallel architectures synthesized in this paper are based on the computation model of distributed memory and distributed control. Several architectural designs have been proposed for the Discrete Wavelet Transform (DWT) [4]-[11]. None of these architectural designs for computing the DWT follows a systematic data dependence and localization analysis of general wavelet-based algorithms, and thus they only serve as particular designs and cannot be extended to other complicated wavelet-based algorithms such as MultiWavelet Transform (MWT)[1,13,14], Wavelet Packet Transform (WPT)[2,15] or Spacial-Frequential Quantization (SFQ) [12]. Using the WPT as a representative example of complex wavelet-based algortihms, this paper fully describes the theory and methodology used in synthesizing parallel architectures for general wavelet-based algorithms.

2 I/O Data Space Modeling of Wavelet-Based Algorithms

The basic equation for any discrete wavelet-based algorithms is generally represented by

$X_{j+1}[t] = \sum_{k \in L} C[k] X_j[Mt - k]$ $(Eq.1)$

where $C[k]$ are taps of a wavelet filter, X_j and X_{j+1} are the *sequence* of input data and output data respectively at the $(j+1)^{th}$ level transform, L is a set that corresponds to the size of the wavelet filter, and M is a constant scalar in the algorithm. Generally, the algorithm is termed as *M-ary wavelet transform* for $M \geq 2$. There are M wavelet filters for M-ary wavelet transform. If X_j and X_{j+1} are scalar data and C is scalar-valued taps of the wavelet filter, the algorithm is a classical scalar wavelet transform; if X_j and X_{j+1} are vector-valued data and C is matrix-valued taps of the multiwavelet filter, it is an MWT. If t and k are scalars, the algorithm is a 1-D transform; if t and k are n-D vectors, it is an n-D transform. Wavelet-based algorithms are multiresolution algorithms, i.e., the output data at a level of transform can be further transformed at the next level.

V.N. Alexandrov et al. (Eds.): ICCS 2001, LNCS 2073, pp. 884–893, 2001.
© Springer-Verlag Berlin Heidelberg 2001

The following concepts are presented for the analysis in Section 3. **Parameter index axis:** The *parameter index axis* of a signal processing algorithm is the index axis for those data to be broadcasted in the algorithm, i.e., the data used in most computations but not generated by computations. As parameters of the computations, the number of them is fixed. **Data index:** The *data index* is the index for the intermediate data, input data, or output data that are generated and/or used by computations. In signal processing algorithms, the input size is variable. **I/O data space:** In an *I/O data space* the indexed data are only possibly the input data, output data or intermediate data, and the parameters of the algorithm are ignored. The intermediate data are viewed as partial inputs or partial outputs for the intermediate computations. **Dependence graph:** In this paper we present the *I/O data space based dependence graph*, where each node in the dependence graph corresponds to a data item, and each edge corresponds to a calculation or a dependence relation between the data used in the calculation and that generated in the calculation. **Wavelet-adjacent field:** In an I/O data space, a *wavelet-adjacent field* is a small domain made up of a group of source data items used by a calculation in Eq. (1). Its size is dependent on the wavelet filter. **Super wavelet-dependence vector:** A *super wavelet-dependence vector* $\overrightarrow{d_{Wb}}$ starts from a wavelet-adjacent field W and ends at the resulted data b. Since the source of the "dependence vector" itself is a domain instead of a single datum, we term such a dependence vector (*corresponding to the calculation in Eq. (1)*) a *super wavelet-dependence vector*. In later analysis the super wavelet-dependence vectors are generally called dependence vectors and treated similarly as traditional dependence vectors. The length of a super wavelet-dependence vector $|\overrightarrow{d_{Wb}}|$ is defined as the Euclidean distance between a and b_c, where a and b_c are arithmetic centers of W and b respectively. **Regular dependence graphs:** in such dependence graphs the length of each dependence vector **d** is a constant value independent of either the input size or the data positions. **Pseudo regular dependence graphs:** in such dependence graphs the dependence vectors can be partitioned into a certain number of groups and in each group the length of dependence vectors is a constant value independent of either the input size or the data positions.

As examples, a wavelet-adjacent field, a super wavelet-dependence vector and the dependence graph for the algorithm of separable 2-D MWT[13][14] in I/O data space are shown in Figure 1 based on these concepts.

3 Nonlinear I/O Data Space Transformations for Regularizing Dependence Graphs

Theorem 3.1: The dependence graphs of wavelet-packet based algorithms (arbitrary expansion of wavelet trees) modeled in I/O data space can always be merged and regularized to a pseudo regular dependence graphs via appropriate nonlinear I/O data space transformations.

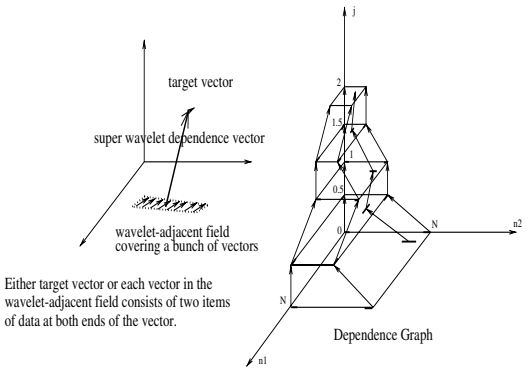


Fig. 1. The wavelet-adjacent field, super dependence vector and the dependence graph for 2-D MWT

Proof:

(1) For algorithms of 1-D transforms: There are M wavelet filters $(f_1, f_2, \cdots, f_M)$ at each level of 1-D M-ary wavelet transform, and each level of transform can decompose a certain subband into M components in wavelet-packet based algorithms. One of the filters (f_1) is for generating coarse component, others for detailed components. Assume M functions $F_i(x) = Mx + i - 1$ for $i = 1, 2, \cdots, M$.

Suppose that a subband $\prod$ is calculated in l levels of wavelet-packet based transform consecutively with wavelet filters $p_1, p_2, \cdots, p_l$, where $p_u = f_i$ for $u = 1, 2, \cdots, l$ and i is any integer $\in [1, M]$. Considering that there are many subbands generated together by l levels of wavelet-packet based transform, and their corresponding dependence graphs should be merged as well as regularized to get a whole dependence graph for the algorithm, the nonlinear I/O data space transformation Γ_1 is presented as follows. Without loosing generality, for the dependence graph corresponding to subband $\prod$, Γ_1 is: $j \longmapsto j; t \longmapsto t$ if $j = 0$; $t \longmapsto P_1(P_2(\cdots(P_j(t))\cdots))$ otherwise, where $P_u = F_i$ if $p_u = f_i$ for $u = 1, 2, \cdots, j$, and $j \leq l$, $i \in [1, M]$. Note that here j corresponds to the level of transform and can be only integers.

Consider another subband $\prod_1$ different from $\prod$ generated in the algorithm. Suppose that $\prod_1$ is calculated in l levels consecutively with wavelet filters $p'_1, p'_2, \cdots, p'_l$, where $p'_u = f_i$ for $u = 1, 2, \cdots, l$ and i is any integer $\in [1, M]$. Since there exits at least one $p'_u \not= p_u$ where $u \in [1, l]$, Γ_1 maps the data for $\prod$ and $\prod_1$ to different positions in the I/O data space. In other words, Γ_1 can combine all dependence graphs of the subbands into a single I/O data space without conflicts.

Consider a dependence vector in the I/O data space corresponding to a calculation of Eq.(1) at data position t_0 at the $(u+1)^{th}$ level of transform,

$X_{u+1}[t_0] = \sum_{k \in L} p_{u+1}[k] X_u[Mt_0 - k]$,

where p_{u+1} represents the wavelet filter used at this level. After the mapping of Γ_1, the calculation changes to

$X_{u+1}[P_1(P_2(\cdots(P_u(P_{u+1}(t_0)))\cdots))] = \sum_{k\in L} p_{u+1}[k]X_u[P_1(P_2(\cdots(P_u(Mt_0 - k))\cdots))]$,

where $P_v = F_i$ if $p_v = f_i$ for $v = 1, 2, \cdots, u+1$, and $i \in [1, M]$. The dependence vector, starting from the wavelet-adjacent field which corresponds to L and is centered at data $X_u[P_1(P_2(\cdots(P_u(Mt_0))\cdots))]$, is targeted to data $X_{u+1}[P_1(P_2(\cdots(P_u(P_{u+1}(t_0)))\cdots))]$. The length of the dependence vector can be resolved by the difference between their coordinates along index j and t. The difference along index j is $|u+1-u| = 1$. The difference along t is $|P_1(P_2(\cdots(P_u(P_{u+1}(t_0)))\cdots)) - P_1(P_2(\cdots(P_u(Mt_0))\cdots))| = |P_1(P_2(\cdots(P_u(Mt_0 + w))\cdots)) - P_1(P_2(\cdots(P_u(Mt_0))\cdots))| = M^u w$, where w is an integer $\in [1, M-1]$. Here the length of the dependence vector is independent of t_0's value, and the number of transform levels and the number of wavelet filters (M) remain constant in the algorithm. In other words, the lengths of the dependence vectors in the I/O data space after the mapping of Γ_1 are bounded and independent of the data positions and input size. Moreover, the dependence vectors can be partitioned into a finite number of groups (according to the possible values of w and u), and the lengths of the dependence vectors in each group are the same. That is, the dependence graphs for 1-D wavelet-packet based algorithm are combined and regularized to be a pseudo regular dependence paragraph via the nonlinear I/O data space transformation Γ_1.

(2) For nonseparable n-D transforms: There are $Q = M^n$ different wavelet filters $(f_1, f_2, \cdots, f_Q)$ at each level of n-D M-ary wavelet transform. Assume Q functions $F_i(x) = Mx + q$, where x and q are n-D vectors. The components of q are $q_1, q_2, \cdots, q_n$, and q_v is an integer $\in [0, M-1]$ for $v = 1, 2, \cdots, n$, and $i = \sum_{u=1}^{n} M^u q_u$. So $i \in [1, Q]$.

Suppose that a subband $\prod$ is calculated in l levels of n-D packet-packet based transform consecutively with wavelet filters $p_1, p_2, \cdots, p_l$, where $p_u = f_i$ for $u = 1, 2, \cdots, l$ and $i \in [1, Q]$. Without loosing generality, for the dependence graph corresponding to subband $\prod$, a nonlinear I/O data space transformation Γ_2 is presented as: $j \mapsto\!\!\!\rightarrow j; t \mapsto\!\!\!\rightarrow t$ if $j = 0$; $t \mapsto\!\!\!\rightarrow P_1(P_2(\cdots(P_j(t))\cdots))$ otherwise, where $P_u = F_i$ if $p_u = f_i$ for $u = 1, 2, \cdots, j$, and $j \le l$, $i \in [1, Q]$. Note that here j corresponds to the level of transform and can be only integers, and t represents n-D vectors.

For other subbands different from $\prod$ generated in the algorithm, since there exits at least one filter used in the calculation of l levels of transform different from that of $\prod$, Γ_2 maps the data of them to different positions. In other words, Γ_2 can combine all dependence graphs of the subbands into a single I/O data space without conflicts.

Similar to the case (1), a calculation corresponding to the dependence vector changes to

$X_{u+1}[P_1(P_2(\cdots(P_u(P_{u+1}(t)))\cdots))] = \sum_{k\in L} p_{u+1}[k]X_u[P_1(P_2(\cdots(P_u(Mt - k))\cdots))]$,

where $P_v = F_i$ if $p_v = f_i$ for $v = 1, 2, \cdots, u+1$, and $i \in [1, Q]$. The difference between the coordinates of the source and the target of the dependence vector along index j is $|u+1-u| = 1$. The difference along t is $P_1(P_2(\cdots(P_u(P_{u+1}(t)))\cdots)) - P_1(P_2(\cdots(P_u(Mt))\cdots)) = P_1(P_2(\cdots(P_u(Mt +$

$w))\cdots)) - P_1(P_2(\cdots(P_u(Mt))\cdots))= M^u w$, where *w is an n-D vector* whose components are integers $\in [1, M-1]$. Thus the length of the dependence vector is independent of t's value. We have the similar conclusion that the lengths of the dependence vectors in the I/O data space after the mapping of Γ_2 are bounded and independent of the data positions and input size, and the dependence vectors can be partitioned into a finite number of groups (according to the possible values of w and u), and the lengths of the dependence vectors in each group are the same.

(3) For separable n-D transforms: The n-D separable transforms are calculated separately and consecutively in every dimension. The index j is drawn in fractional numbers to represent the intermediate calculations in each level of transform. In the $(s+1)^{th}$ level (where s is an non-negative integer) of a separable n-D wavelet transforms, we have (n-1) intermediate I/O data planes $j = s + 1/n$, $j = s + 2/n$, $\cdots$, $j = s + (n-1)/n$ between the planes $j = s$ and $j = s + 1$. In the calculations for every dimension, there are M wavelet filters $f_1, f_2, \cdots, f_M$, and a subband may be decomposed into M components on each dimension. So after each level of transform, a subband can be decomposed into M^n components. In addition, we assume M functions $F_v(x) = Mx + v - 1$ for $v = 1, 2, \cdots, M$.

Suppose that a certain subband $\prod$ is calculated in l levels of n-D separable wavelet-packet based transform consecutively with wavelet filters $p_{1,1}, p_{1,2}, \cdots, p_{1,n}, p_{2,1}, \cdots, p_{2,n}, \cdots, p_{l,n}$, where $p_{u,i} = f_v$ for $u = 1, 2, \cdots, l$ and $i = 1, 2, \cdots, n$, and $v \in [1, M]$. $p_{u,i}$ represents the wavelet filter used for the calculation of the u^{th} level transform on the i^{th} dimension in generating $\prod$. In order to regularize the dependence graphs, we present the nonlinear I/O data space transformation Γ_3 as follows. Without loosing generality, for the dependence graph corresponding to subband $\prod$, Γ_3 is: $j \longmapsto j$, $t_i \longmapsto t_i$ $(i = 1, 2, \cdots, n)$ if $j = 0$; $t_i \longmapsto P_{1,i}(P_{2,i}(\cdots(P_{s+1,i}(t_i))\cdots))$ otherwise, with $j \in [s + i/n, s + 1 + i/n)$, s being an integer $\in [0, l-1]$, $P_{u,i} = F_v$ for $p_{u,i} = f_v$ $(u = 1, 2, \cdots, s+1;\ i = 1, 2, \cdots, n;$ and $v \in [1, M])$.

For other subbands different from $\prod$ generated in the algorithm, since there exits at least one filter used in the calculation of l levels of transform different from that of $\prod$, Γ_3 maps the data of them to different positions.

The calculation of wavelet transform for the i^{th} dimension at the $(s+1)^{th}$ level of transform for $\prod$,

$\sum_{k_i \in L_i} p_{s+1,i}[k_i] \sum_{k_{i+1} \in L_{i+1}} p_{s+1,i+1}[k_{i+1}] \cdots \sum_{k_n \in L_n} p_{s+1,n}[k_n] X_{s+(i-1)/n}[t_1, t_2, \cdots, Mt_i - k_i, Mt_{i+1} - k_{i+1}, \cdots, Mt_n - k_n]$,
$= \sum_{k_{i+1} \in L_{i+1}} p_{s+1,i+1}[k_{i+1}] \cdots \sum_{k_n \in L_n} p_{s+1,n}[k_n] X_{s+i/n}[t_1, t_2, \cdots, t_i, Mt_{i+1} - k_{i+1}, \cdots, Mt_n - k_n]$,
changes to calculating

$\sum_{k_i \in L_i} p_{s+1,i}[k_i] \sum_{k_{i+1} \in L_{i+1}} p_{s+1,i+1}[k_{i+1}] \cdots \sum_{k_n \in L_n} p_{s+1,n}[k_n] X_{s+(i-1)/n}[P_{1,1}(P_{2,1}(\cdots(P_{s+1,1}(t_1))\cdots)), P_{1,2}(P_{2,2}(\cdots(P_{s+1,2}(t_2))\cdots)), \cdots, P_{1,i}(P_{2,i}(\cdots(P_{s,i}(Mt_i - k_i))\cdots)), P_{1,i+1}(P_{2,i+1}(\cdots(P_{s,i+1}(Mt_{i+1} - k_{i+1}))\cdots)), \cdots, P_{1,n}(P_{2,n}(\cdots(P_{s,n}(Mt_n - k_n))\cdots))]$,
$= \sum_{k_{i+1} \in L_{i+1}} p_{s+1,i+1}[k_{i+1}] \cdots \sum_{k_n \in L_n} p_{s+1,n}[k_n]\ X_{s+i/n}[P_{1,1}(P_{2,1}(\cdots(P_{s+1,1}(t_1))\cdots)), P_{1,2}(P_{2,2}(\cdots(P_{s+1,2}(t_2))\cdots)), \cdots, P_{1,i}(P_{2,i}(\cdots(P_{s+1,i}(t_i))\cdots)), P_{1,i+1}(P_{2,i+1}(\cdots$

$(P_{s,i+1}(Mt_{i+1}-k_{i+1}))\cdots)),\cdots,P_{1,n}(P_{2,n}(\cdots(P_{s,n}(Mt_n-k_n))\cdots))]$ after the mapping of Γ_3, where $P_{u,i}=F_v$ if $p_{u,i}=f_v$ for $u=1,2,\cdots,s+1$; $i=1,2,\cdots,n$; and $v\in[1,M]$.

Thus after the transformation Γ_3, the dependence vector, starting from the wavelet-adjacent field which corresponds to L_i and is centered at data $X_{s+(i-1)/n}[P_{1,1}(P_{2,1}\ (\cdots(P_{s+1,1}(t_1))\cdots)),\ P_{1,2}(P_{2,2}(\cdots\ (P_{s+1,2}(t_2))\cdots)),$ $\cdots,P_{1,i}(P_{2,i}\ \ (\cdots(P_{s,i}(Mt_i))\ \ \cdots)),P_{1,i+1}(P_{2,i+1}\ \ (\cdots(P_{s,i+1}\ \ (Mt_{i+1}-\ \ k_{i+1}))$ $\cdots)),$ $\cdots,P_{1,n}\ (P_{2,n}(\cdots(P_{s,n}\ (Mt_n-k_n))\cdots))]$, is targeted to data $X_{s+i/n}[P_{1,1}(P_{2,1}(\cdots(P_{s+1,1}(t_1))\cdots)),P_{1,2}(P_{2,2}(\cdots(P_{s+1,2}(t_2))\cdots)),\cdots,P_{1,i}$ $(P_{2,i}\ (\cdots(P_{s+1,i}(t_i))\cdots)),P_{1,i+1}(P_{2,i+1}(\cdots(P_{s,i+1}(Mt_{i+1}-k_{i+1}))\cdots)),\cdots,P_{1,n}$ $(P_{2,n}(\cdots(P_{s,n}(Mt_n\ -\ k_n))\cdots))]$. The difference between the coordinates of the target and the source of the dependence vector along index j is $|(s+i/n)-(s+(i-1)/n)|=1/n$. The difference along t is $P_{1,i}(P_{2,i}(\cdots(P_{s,i}(P_{s+1,i}(t_i)))\cdots))-P_{1,i}(P_{2,i}(\cdots(P_{s,i}(Mt_i))\cdots))$ $=\ P_{1,i}(P_{2,i}(\cdots(P_{s,i}(Mt_i\ +\ w))\cdots))\ -\ P_{1,i}(P_{2,i}(\cdots(P_{s,i}(Mt_i))\cdots))=\ M^sw$, where w is an integer $\in[1,M-1]$. Thus the length of the dependence vector is independent of t's value. We have the similar conclusion that the lengths of the dependence vectors in the I/O data space after the mapping of Γ_3 are bounded and independent of the data positions and the input size, and the dependence vectors can be partitioned into a finite number of groups (according to the possible values of w and s), and the lengths of the dependence vectors in each group are the same. □

4 Design Example: Synthesis of 2-D WPT by Exploiting Inter-iteration Parallelism

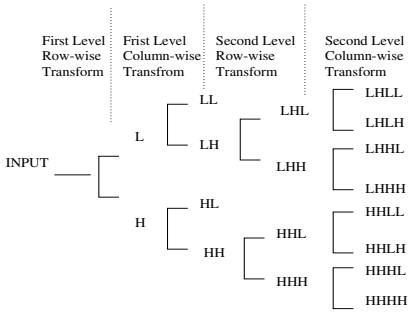


Fig. 2. An instance of arbitrary wavelet tree expansion in the algorithm of 2-D WPT

The recursive separable 2-D WPT is illustrated in Figure 2 and the equations are as the followings

$C^{(j+1,4i)}[n_1,n_2] = \sum_{k1}\sum_{k2} h[k_1]h[k_2] \times C^{j,i}[2n_1-k_1, 2n_2-k_2]$
$C^{(j+1,4i+1)}[n_1,n_2] = \sum_{k1}\sum_{k2} h[k_1]g[k_2] \times C^{j,i}[2n_1-k_1, 2n_2-k_2]$
$C^{(j+1,4i+2)}[n_1,n_2] = \sum_{k1}\sum_{k2} g[k_1]h[k_2] \times C^{j,i}[2n_1-k_1, 2n_2-k_2]$
$C^{(j+1,4i+3)}[n_1,n_2] = \sum_{k1}\sum_{k2} g[k_1]g[k_2] \times C^{j,i}[2n_1-k_1, 2n_2-k_2]$
where $C^{(j,i)}[n_1,n_2]$ means the datum at the position of n_1^{th} row, n_2^{th} column in the i^{th} subband in transform level j, h and g are low- and high-pass wavelet filters. $C^{(0,0)}$ is the input image, L_0 is the wavelet filter length, J is the highest transform level, and N^2 is the size of the input image.

In Figure 2, the label for subband $C^{(j,i)}$ generated in the j^{th} $(1 \le j \le J)$ level of WPT is given as a combination of H's and L's, which represent a binary number if we refer to H as "1" and L as "0". This binary number is equal to i. For example, subband $C^{(2,14)}$ is labeled as $HHHL$ in Figure 2, or "1110" (14 in decimal). Note that some components in a transform level may not be recursively decomposed in the next level WPT transform. According to Γ_3 in Section 3, the nonlinear I/O index space transformation to merge all dependence graphs for generated subbands (as in Figure 2) is: (1) for the $p_1p_2p_3.....p_{2m-1}p_{2m}$ subband in Figure 2 (result of the m^{th} level 2-D WPT), $n_1 \mapsto\!\!\!\!\longrightarrow P_2(P_4(...(P_{2m}(n_1))...))$; $n_2 \mapsto\!\!\!\!\longrightarrow P_1(P_3(...(P_{2m-1}(n_2))...))$; $j = m$; (2) for the $p_1p_2p_3.....p_{2m}p_{2m+1}$ subband in Figure 2 (intermediate result in the $(m+1)^{th}$ level 2-D WT), $n_1 \mapsto\!\!\!\!\longrightarrow P_2(P_4(...(P_{2m}(n_1))...))$; $n_2 \mapsto\!\!\!\!\longrightarrow P_1(P_3(...(P_{2m+1}(n_2))...))$; $j = m + \frac{1}{2}$, where we assume that function $P_k(x)$ is $Low(x) = 2x$ if p_k is "L"; or $P_k(x)$ is $High(x) = 2x + 1$ if p_k is "H" for $k = 0, 1, 2, \cdots, 2m$. Figure 3 shows the result of the nonlinear transformation in plane *j=2* in the I/O data space.

In this section, we propose the parallel computing of 2-D WPT by exploiting the inter-iteration parallelism *based on the regularized and merged dependence graphs* via this nonlinear I/O data space transformation. The input, pixels of a 2-D image signal are assumed to be fed to multi-processors in parallel. The following concepts are adopted in the rest of this section.

Processor assignment: In this paper *processor assignment* is taken equivalently as *I/O data space segmentation*, where the I/O data space is segmented into subspaces, and the computations corresponding to the super dependence vectors in each segmented subspace are assigned to a processor.

Boundary dependence vectors vs. central dependence vectors: After segmenting the I/O data space, those dependence vectors lying in more than one subspaces of the I/O index space are called *boundary dependence vectors*, otherwise *central dependence vectors*.

Computation scheduling and permissible scheduling: the processor assignment is accompanied by a *computation scheduling* scheme, which specifies the order of the calculations in all the processors. A *permissible schedule* must satisfy two conditions: 1) the *inherently sequential* computations cannot be scheduled to the same time, i.e., the schedule cannot contradict the dependence graph; 2) no more than one computations can be performed in a processor at the same time.

Boundary computation: A processor's *boundary computations* are those performed in this processor and necessitating that the result of the computations be sent out to other processors.

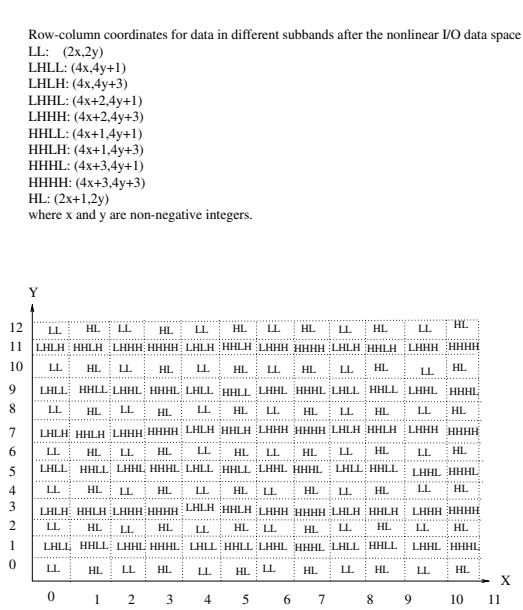


Fig. 3. The data redistribution based on the I/O data space transformation for the 2-D WPT algorithm shown as in Figure 2

For briefing communication network and concentrating on the computation scheduling within processors based on the I/O data space transformation, we assume mesh-like processor (or PE) array to be used to implement the algorithm. For the purpose of minimizing data communication intensity, the total number of boundary dependence vectors is made as small as possible after the I/O data space segmentation. Thus, we segment the I/O data space in the direction of most dependence vectors, i.e., along the direction parallel with j axis. The shape of dependence graphs in each segmented subspace has the similarity as that in the whole I/O data space, but with different boundary dependence vectors generated in the segmentation. For the purpose of load balance among processors, all the subspaces are supposed to have the same size after the segmentation. Thus, we have partitioned the I/O data space into p^2 subspaces as in Figure 4, where p^2 is the number of parallel processors (or PE's).

To minimize the data communication intensity or the requirement on the network bandwidth, the boundary computations in a processor are scheduled as far away as possible in timing. A simpler explanation for this is that the result of a boundary computation is sent as soon as calculated and it is alright if the communication is completed before the result of the next boundary computation is generated and sent in the sense that the communication conflicts are avoided. In other words, to maximize the intervals between the boundary computations, each processor takes turns to execute one boundary computation and R non-boundary computations which involve boundary dependence vectors and central dependence vectors respectively, where R is the ratio of the number

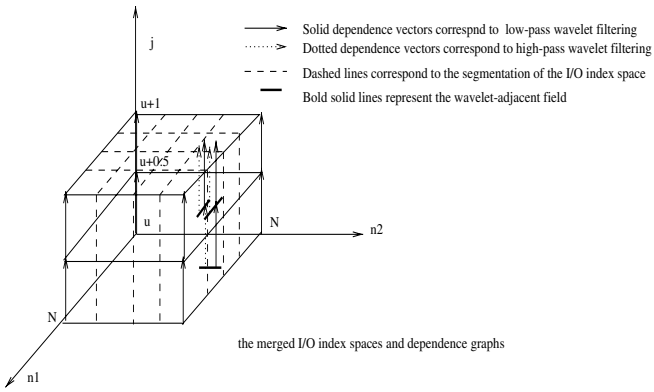


Fig. 4. The segmentation of I/O data space

of central dependence vectors to the number of boundary dependence vectors in the processor.

5 Conclusions

This paper has demonstrated that data dependence analysis provides the basis for the synthesis of parallel architectural solutions for general wavelet-based algorithms and serves as a theoretical foundation for exploiting properties. Extracting the common features of computation locality and multirate signal processing within the wavelet-based algorithms, this paper contributes to data dependence and localization analysis based on a new concept — I/O data space analysis which leads to simplified structures of dependence graphs, and nonlinear I/O data space transformations for generalized high-level architectural synthesis of wavelet-based algorithms.

References

1. M.. Cotronei, L. B. Montefusco and L. Puccio, Multiwavelet analysis and signal processing, *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, Vol. 45, Aug. 1998, pp. 970 -987.
2. F. G. Meyer, A. Z. Averbuch and J. O. Stromberg, Fast adaptive wavelet packet image compression, *IEEE Transactions on Image Processing*, Vol. 9, May 2000, pp. 792 -800.
3. J. Fridman and E. S. Manolakos, Discrete wavelet transform: data dependence analysis and synthesis of distributed memory and control array architectures, *IEEE Transactions on Signal Processing*, Vol. 45, May 1997, pp. 1291 -1308.
4. H. Sava, M. Fleury, A. C. Downton and A. F. Clark, Parallel pipeline implementation of wavelet transforms, *IEE Proceedings on Vision, Image and Signal Processing*, Vol. 144, Dec. 1997, pp. 355 -360.

5. K. K. Parhi and T. Nishitani, VLSI architectures for discrete wavelet transforms, *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, Vol. 1, June 1993, pp. 191 -202.
6. A. Grzeszczak, M. K. Mandal and S. Panchanathan, VLSI implementation of discrete wavelet transform, *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, Vol. 4, Dec. 1996, pp. 421 -433.
7. Seung-Kwon Pack and Lee-Sup Kim, 2D DWT VLSI architecture for wavelet image processing, *Electronics Letters*, Vol. 34, March 1998, pp. 537 -538.
8. G. Knowles, VLSI architecture for the discrete wavelet transform, *Electronics Letters*, Vol. 26, 19 July 1990, pp. 1184 -1185.
9. M. Vishwanath, R. M. Owens and M. J. Irwin, VLSI architectures for the discrete wavelet transform, *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, Vol. 42, May 1995, pp. 305 -316.
10. T. C. Denk and K. K. Parhi, VLSI architectures for lattice structure based orthonormal discrete wavelet transforms, *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, Vol. 44, Feb. 1997, pp. 129 -132.
11. C. Chakrabarti and M. Vishwanath, Efficient realizations of the discrete and continuous wavelet transforms: from single chip implementations to mappings on SIMD array computers, *IEEE Transactions on Signal Processing*, Vol. 43, March 1995, pp. 759 -771.
12. Zixiang Xiong, K. Ramchandran and M.T.Orchard, Wavelet packet image coding using space-frequency quantization, *IEEE Transactions on Image Processing*, Vol. 7, June 1998, pp. 892 -898.
13. M. Cotronei, D. Lazzaro, L. B. Montefusco and L. Puccio, Image compression through embedded multiwavelet transform coding, *IEEE Transactions on Image Processing*, Vol. 9, Feb. 2000, pp. 184 -189.
14. Gang Lin and Ze-Min Liu, The application of multiwavelet transform to image coding, *IEEE Transactions on Image Processing*, Vol. 9, Feb. 2000, pp. 270 -273.
15. F. Kurth and M. Clausen, Filter bank tree and M-band wavelet packet algorithms in audio signal processing, *IEEE Transactions on Signal Processing*, Vol. 47, Feb. 1999, pp. 549 -554.

Solvable Map Method for Integrating Nonlinear Hamiltonian Systems

Govindan Rangarajan and Minita Sachidanand

Department of Mathematics
Indian Institute of Science
Bangalore 560 012, India
rangaraj@math.iisc.ernet.in

Abstract. Conventional numerical integration algorithms can not be used for long term stability studies of complicated nonlinear Hamiltonian systems since they do not preserve the symplectic structure of the system. Further, they can be very slow even if supercomputers are used. In this paper, we study the symplectic integration algorithm using solvable maps which is both fast and accurate and extend it to six dimensions. This extension enables single particle studies using all three degrees of freedom.

1 Introduction

Consider a complicated Hamiltonian system that is non-integrable. Suppose we are interested in the long-term stability of this dynamical system. Since the system is assumed to be non-integrable, it is very difficult to give stability criteria in an analytic form. A possible solution is to numerically follow the trajectories of the particles through the system for a large number of iterations, a process which goes by the name tracking. One could then infer the stability of motion in the system by analyzing these tracking results. However, in long term integration of these systems, it is important to preserve the Hamiltonian nature of the system at every integration step. Otherwise, one can get spurious damping or even chaotic behaviour which is not present in the original system. Such problems becomes accentuated when long term integration is performed. This can then obviously lead to wrong predictions regarding the long-term stability of the Hamiltonian system being studied.

The most straightforward method that can be used to perform long term tracking is numerical integration. However, a very short time step has to be used to reduce the spurious behaviour that occurs because of the non-symplectic integration. This makes this method so slow that it is impractical to study the long term behaviour of very complicated systems (like the Large Hadron Collider) even if supercomputers are used. Therefore, we need a method that is both fast and accurate.

Several symplectic integration methods have been discussed in literature [1,2, 3,4,5,6,7,8,9,10,11]. Methods using Lie algebraic perturbation theory which give

V.N. Alexandrov et al. (Eds.): ICCS 2001, LNCS 2073, pp. 894–903, 2001.
© Springer-Verlag Berlin Heidelberg 2001

maps where the final values of variables are explicit functions of initial ones offer several advantages. Further, since the whole complicated system can be represented by a single symplectic map, there can be a substantial saving in computation time as compared to non-map based methods. In this paper, we extend an alternate symplectic integration method using solvable maps [12] to six dimensions. This extension is necessary for single particle studies using all three degrees of freedom. The above method has been shown to give good results in lower dimensions [12].

2 Preliminaries

Let $z = (q_1, q_2, \ldots, q_n, p_1, p_2, \ldots, p_n)$ denote the $2n$ dimensional phase space variables. Denoting the initial and final locations of the particle by z^i and z^f respectively, the time evolution of a Hamiltonian system $H(z)$ can be described as a symplectic map $\mathcal{M}$ [13]:

$$z^f = \mathcal{M} z^i \,. \tag{1}$$

The symplectic map is related to the Hamiltonian $H(z)$ by the equation:

$$\frac{d\mathcal{M}}{dt} = \mathcal{M} : -H(z_0), \tag{2}$$

where z_0 is z at time $t = 0$.

Using the Dragt-Finn factorization theorem [13], the symplectic map $\mathcal{M}$ can be factorized as follows.

$$\mathcal{M} = \hat{M} \exp(: f_3(z) :) \cdots \exp(: f_m(z) :) \cdots \,. \tag{3}$$

Here $\hat{M}$ represents the linear part of the map and $f_m(z)$ is a homogeneous polynomial of degree m in z determined uniquely by the factorization theorem. Further $: f(z) :$ is the Lie operator corresponding to the function $f(z)$ and is defined by

$$: f(z) : \; g(z) = [f(z), g(z)]. \tag{4}$$

Here $g(z)$ denotes another phase space function and $[f(z), g(z)]$ denotes the usual Poisson bracket of the functions $f(z)$ and $g(z)$. The exponential of a Lie operator is called a Lie transformation and is given as

$$\exp(: f(z) :) = \sum_{n=0}^{\infty} \frac{: f(z) :^n}{n!}. \tag{5}$$

The map $\mathcal{M}$ involves an infinite number of Lie transformations and hence for any practical computation, we have to truncate $\mathcal{M}$ after a finite number of Lie transformations.

$$\mathcal{M} = \hat{M}\, e^{:f_3:}\, e^{:f_4:} \cdots e^{:f_m:} \,. \tag{6}$$

Each one of the Lie transformations in the above equation is a symplectic map and hence the map can be truncated at any order without losing symplecticity.

The product $e^{:f_3:}\, e^{:f_4:} \cdots e^{:f_m:}$ in Eq. (6) give the nonlinear part of $\mathcal{M}$. Each of these Lie transformations gives an infinite number of terms when acting on phase space variables. Since we can not evaluate an infinite number of terms in any algorithm, we need to overcome this problem in some manner. The most straightforward method is to truncate the Taylor series expansion of each Lie transformation, after a finite number terms. This however violates the symplectic condition. Though this method is justifiable in short term tracking, it does not work well in long term tracking as the non-symplecticity can lead to spurious damping or even chaotic behaviour which is not present in the original system [10]. Therefore, we refactorize $\mathcal{M}$ in terms of simpler symplectic maps that can be evaluated both exactly and quickly. We achieve this refactorization through the so-called "solvable maps" [11,12].

3 Solvable Map Method

Solvable maps are generalizations of Cremona maps. The class of Cremona maps includes only those symplectic maps for which the Taylor series expansion terminates when acting on phase space coordinates. The class of solvable maps also includes those symplectic maps for which the Taylor series expansion can be summed up explicitly. One simple example of such a map is $\exp(: aq_1^{l+2} + bq_1^{l+1}p_1 :)$ [12].

The basic idea behind the solvable map method is to represent each nonlinear factor $\exp(: f_m :)$ in Eq. (6) as a product of solvable maps, i.e.,

$$\exp(: f_m :) = \exp(: h_1 :)\, \exp(: h_2 :) \cdots \exp(: h_n :)\,, \text{ for } \; m \geq 3\,. \tag{7}$$

We also ensure that the number of solvable maps is a minimum. For simplicity, we restrict ourselves to a general fourth order symplectic map in six dimensions.

We first set up a notation that will facilitate the indexing of monomials in six phase space variables. Let P_j denote the following basis monomial of degree m in six phase-space variables

$$P_j^{(m)} = q_1^{r_1} p_1^{r_2} q_2^{r_3} p_2^{r_4} q_3^{r_5} p_3^{r_6}\,, \tag{8}$$

where $1 \leq r_i \leq m, \quad r_1 + r_2 + \ldots + r_6 = m$. For this six dimensional case, the number of independent monomials of degree 3 is 56 and the number of independent monomials of degree 4 is 126.

Consider a nonlinear Hamiltonian system in six phase space dimensions. Performing a Taylor expansion around the origin in the spirit of perturbation theory, the Hamiltonian $H(z)$ can be written as (up to fourth order in the phase space variables z)

$$H(z) = H_2(z) + H_3(z) + H_4(z). \tag{9}$$

Here $H_m(z)$ is a homogeneous polynomial of degree m in z. Concentrating on the nonlinear part for the moment, H_3 and H_4 can be written as a linear combination

of the basis monomials introduced above:

$$H_3(z) = \gamma_1 P_1(z) + \gamma_2 P_2(z) + \cdots + \gamma_{56} P_{56}(z), \tag{10}$$
$$H_4(z) = \gamma_{57} P_{57}(z) + \gamma_{58} P_{58}(z) + \cdots + \gamma_{182} P_{182}(z). \tag{11}$$

Now we express the time evolution of the Hamiltonian system $H(z)$ as a symplectic map $\mathcal{M}_4$ (truncated at order 4) using the Dragt-Finn factorization:

$$\mathcal{M}_4 = \hat{M} \exp(:\, f_3 :)\ \exp(:\, f_4 :)\,, \tag{12}$$

where $\hat{M}$ is obtained from the linear part $H_2(z)$ of the Hamiltonian $H(z)$. Using the basis monomials, f_3 and f_4 can be represented as:

$$f_3(z) = \alpha_1 P_1(z) + \alpha_2 P_2(z) + \cdots + \alpha_{56} P_{56}(z), \tag{13}$$
$$f_4(z) = \alpha_{57} P_{57}(z) + \alpha_{58} P_{58}(z) + \cdots + \alpha_{182} P_{182}(z). \tag{14}$$

It should be noted that there is a specific algorithm for determining the coefficients α_i in terms of the coefficients γ_i and the linear part $H_2(z)$. For a specific Hamiltonian, many of the coefficients α_i could be zero.

We now refactorize the symplectic map $\mathcal{M}_4$ representing the original Hamiltonian system in terms of solvable maps as follows (for further details see [14]):

$$\mathcal{M} = \hat{M}\, e^{:h_1:} e^{:h_2:} \ldots e^{:h_{20}:}\,, \tag{15}$$

where

$$\begin{aligned}
h_1 = {} & q_1^4\, [\,\beta_{57}\,] + q_1^3\, [\,\beta_1 + \beta_{59}\, q_2 + \beta_{61}\, q_3\,] + \\
& q_1^2\, \big[\,\beta_3\, q_2 + \beta_5\, q_3 + \beta_{68}\, q_2^2 + \beta_{70}\, q_2\, q_3 + \beta_{75}\, q_3^2\,\big] + \\
& q_1\, \big[\,\beta_{12}\, q_2^2 + \beta_{14}\, q_2\, q_3 + \beta_{19}\, q_3^2 + \beta_{93}\, q_2^3 + \beta_{95}\, q_2^2\, q_3 + \beta_{100}\, q_2\, q_3^2 + \\
& \qquad \beta_{109}\, q_3^3\,\big] + \\
& \big[\,\beta_{39}\, q_2^2\, q_3 + \beta_{44}\, q_2\, q_3^2 + \beta_{150}\, q_2^3\, q_3 + \beta_{155}\, q_2^2\, q_3^2 + \beta_{164}\, q_2\, q_3^3\,\big] + \\
& p_1\, \big[\,\beta_{27}\, q_2^2 + \beta_{29}\, q_2\, q_3 + \beta_{34}\, q_3^2 + \beta_{128}\, q_2^3 + \beta_{130}\, q_2^2\, q_3 + \\
& \qquad \beta_{135}\, q_2\, q_3^2 + \beta_{144}\, q_3^3\,\big] + \\
& q_1\, p_1\, \big[\,\beta_8\, q_2 + \beta_{10}\, q_3 + \beta_{83}\, q_2^2 + \beta_{85} q_2\, q_3 + \beta_{90}\, q_3^2\,\big]\,,
\end{aligned}$$

$$\begin{aligned}
h_2 = {} & p_1^4\, [\beta_{113}\,] + p_1^3\, [\,\beta_{22} + \beta_{115}\, p_2 \ + \ \beta_{117}\, p_3\,] + \\
& p_1^2\, \big[\,\beta_{24}\, p_2 + \beta_{26}\, p_3 + \beta_{122}\, p_2^2 + \beta_{124}\, p_2\, p_3 + \beta_{127}\, p_3^2\,\big] + \\
& p_1\, \big[\,\beta_{31}\, p_2^2 + \beta_{33}\, p_2\, p_3 + \beta_{36}\, p_3^2 + \beta_{138}\, p_2^3 + \beta_{140}\, p_2^2\, p_3 + \\
& \qquad \beta_{143}\, p_2\, p_3^2 + \beta_{147}\, p_3^3\,\big] + \\
& \big[\,\beta_{49}\, p_2^2\, p_3 + \beta_{52}\, p_2\, p_3^2 + \beta_{170}\, p_2^3\, p_3 + \beta_{173}\, p_2^2\, p_3^2 + \beta_{177}\, p_2\, p_3^3\,\big] + \\
& q_1\, \big[\,\beta_{16}\, p_2^2 + \beta_{18}\, p_2\, p_3 + \beta_{21}\, p_3^2 + \beta_{103}\, p_2^3 + \beta_{105}\, p_2^2\, p_3 + \\
& \qquad \beta_{108}\, p_2\, p_3^2 + \beta_{112}\, p_3^3\,\big] + \\
& q_1\, p_1\, \big[\,\beta_9\, p_2 + \beta_{11}\, p_3 + \beta_{87}\, p_2^2 + \beta_{89}\, p_2\, p_3 + \beta_{92}\, p_3^2\,\big]\,,
\end{aligned}$$

$$\begin{aligned}
h_3 = {} & q_2^4\, [\,\beta_{148}\,] + q_2^3\, [\,\beta_{37} + \beta_{151}\, p_3\,] + \\
& q_2^2\, \big[\,\beta_{40}\, p_3 + \beta_{118}\, p_1^2 + \beta_{131}\, p_1\, p_3 + \beta_{157}\, p_3^2\,\big] +
\end{aligned}$$

$$
\begin{aligned}
& q_2 \, \big[\, \beta_{23} \, p_1^2 + \beta_{30} \, p_1 \, p_3 + \beta_{46} \, p_3^2 + \beta_{114} \, p_1^3 + \beta_{121} \, p_1^2 \, p_3 + \\
& \qquad \beta_{137} \, p_1 \, p_3^2 + \beta_{167} \, p_3^3 \, \big] + \\
& q_2 \, p_2 \, \big[\, \beta_{28} \, p_1 + \beta_{43} \, p_3 + \beta_{119} \, p_1^2 + \beta_{134} \, p_1 \, p_3 + \beta_{163} \, p_3^2 \, \big] \, ,
\end{aligned}
$$

$$
\begin{aligned}
h_4 = \; & p_2^4 \, [\, \beta_{168} \,] + p_2^3 \, [\, \beta_{47} + \beta_{169} \, q_3 \,] + \\
& p_2^2 \, \big[\, \beta_{48} \, q_3 + \beta_{72} \, q_1^2 + \beta_{104} \, q_1 \, q_3 + \beta_{171} \, q_3^2 \, \big] + \\
& p_2 \, \big[\, \beta_4 \, q_1^2 + \beta_{17} \, q_1 \, q_3 + \beta_{50} \, q_3^2 + \beta_{60} \, q_1^3 + \beta_{73} \, q_1^2 \, q_3 + \beta_{106} \, q_1 \, q_3^2 + \beta_{174} \, q_3^3 \, \big] + \\
& q_2 \, p_2 \, \big[\, \beta_{13} \, q_1 + \beta_{42} \, q_3 + \beta_{69} \, q_1^2 + \beta_{98} \, q_1 \, q_3 + \beta_{161} \, q_3^2 \, \big] \, ,
\end{aligned}
$$

$$
\begin{aligned}
h_5 = \; & q_3^4 \, [\, \beta_{178} \,] + q_3^3 \, [\, \beta_{53} \,] + q_3^2 \, \big[\, \beta_{125} \, p_1^2 + \beta_{141} \, p_1 \, p_2 \, \big] + \\
& q_3 \, \big[\, \beta_{25} \, p_1^2 + \beta_{32} \, p_1 \, p_2 + \beta_{116} \, p_1^3 + \beta_{123} \, p_1^2 \, p_2 + \beta_{139} \, p_1 \, p_2^2 \, \big] + \\
& q_3 \, p_3 \, \big[\, \beta_{35} \, p_1 + \beta_{51} \, p_2 + \beta_{126} \, p_1^2 + \beta_{142} \, p_1 \, p_2 + \beta_{172} \, p_2^2 \, \big] \, ,
\end{aligned}
$$

$$
\begin{aligned}
h_6 = \; & p_3^4 \, [\, \beta_{182} \,] + p_3^3 \, [\, \beta_{56} \,] + p_3^2 \, \big[\, \beta_{77} \, q_1^2 + \beta_{102} \, q_1 \, q_2 \, \big] + \\
& p_3 \, \big[\, \beta_6 \, q_1^2 + \beta_{15} \, q_1 \, q_2 + \beta_{62} \, q_1^3 + \beta_{71} \, q_1^2 \, q_2 + \beta_{96} \, q_1 \, q_2^2 \, \big] + \\
& q_3 \, p_3 \, \big[\, \beta_{20} \, q_1 + \beta_{45} \, q_2 + \beta_{76} \, q_1^2 + \beta_{101} \, q_1 \, q_2 + \beta_{156} \, q_2^2 \, \big] \, ,
\end{aligned}
$$

$$
h_7 = \beta_{74} \, q_1^2 \, p_2 \, p_3 \, ,
$$

$$
h_8 = \beta_{120} \, p_1^2 \, q_2 \, q_3 \, ,
$$

$$
h_9 = q_1^2 \, p_1 \, [\, \beta_2 + \beta_{64} \, q_2 + \beta_{65} \, p_2 \; + \; \beta_{66} \, q_3 + \beta_{67} \, p_3 \,] \, ,
$$

$$
h_{10} = \; q_1 \, p_1^2 \, [\, \beta_7 + \beta_{79} \, q_2 + \beta_{80} \, p_2 \; + \beta_{81} \, q_3 + \beta_{82} \, p_3 \,] \, ,
$$

$$
h_{11} = \; q_2^2 \, p_2 \, [\, \beta_{38} + \beta_{94} \, q_1 + \beta_{129} \, p_1 \; + \beta_{153} \, q_3 + \beta_{154} \, p_3 \,] \, ,
$$

$$
h_{12} = \; q_2 \, p_2^2 \, [\, \beta_{41} + \beta_{97} \, q_1 + \beta_{132} \, p_1 \; + \beta_{159} \, q_3 + \beta_{160} \, p_3 \,] \, ,
$$

$$
h_{13} = \; q_3^2 \, p_3 \, [\, \beta_{54} + \beta_{110} \, q_1 + \beta_{145} \, p_1 \; + \beta_{165} \, q_2 + \beta_{175} \, p_2 \,] \, ,
$$

$$
h_{14} = \; q_3 \, p_3^2 \, [\, \beta_{55} + \beta_{111} \, q_1 + \beta_{146} \, p_1 \; + \beta_{166} \, q_2 + \beta_{176} \, p_2 \,] \, ,
$$

$$
h_{15} = \beta_{58} \, q_1^3 \, p_1 + \beta_{149} \, q_2^3 \, p_2 + \beta_{179} \, q_3^3 \, p_3 \, ,
$$

$$
h_{16} = \beta_{78} \, q_1 \, p_1^3 + \beta_{158} \, q_2 \, p_2^3 + \beta_{181} \, q_3 \, p_3^3 \, ,
$$

$$
h_{17} = \beta_{86} \, q_1 \, p_1 \, q_2 \, p_3 + \beta_{88} \, q_1 \, p_1 \, p_2 \, q_3 \, ,
$$

$$h_{18} = \beta_{99}\, q_2\, p_2\, q_1\, p_3 + \beta_{133}\, q_2\, p_2\, p_1\, q_3\,,$$

$$h_{19} = \beta_{107}\, q_3\, p_3\, q_1\, p_2 + \beta_{136}\, q_3\, p_3\, q_2\, p_1\,,$$

$$h_{20} = \beta_{63}\, q_1^2\, p_1^2 + \beta_{84}\, q_1\, p_1\, q_2\, p_2 + \beta_{91}\, q_1\, p_1\, q_3\, p_3 + \beta_{152}\, q_2^2\, p_2^2 + \beta_{162}\, q_2\, p_2\, q_3\, p_3 + \beta_{180}\, q_3^2\, p_3^2\,.$$

Here β_i's are functions of α_i's. They can be easily determined by comparing the original factorization for $\mathcal{M}_4$ in terms of f_i with the solvable map refactorization in terms of h_i order by order and using the Campbell-Baker-Hausdorff theorem [15]. Thus given γ_i's parameterizing the original Hamiltonian system $H(z)$, β_i's can be readily determined [14].

We will now explicitly obtain the action of some of the above maps $e^{:h_i:}$ on the phase space variables in a closed form. This will also prove that these maps are solvable maps. A similar analysis can be done for the remaining maps [14]. Consider the action of $e^{:h_i:}$ on z:

$$z' = e^{:h_i:} z. \tag{16}$$

From Eq. (2) we see that this action is equivalent to first integrating the equations of motion for the Hamiltonian $h_i(z)$ from time $t = 0$ to time $t = -1$ and then identifying the initial values z_0 (at $t = 0$) with z and the final values (at $t = -1$) for z with z'.

First, we consider the action of $e^{:h_7:}$ on the phase space variables. Using the above procedure, this action can be easily obtained and is given as follows:

$$\begin{aligned} q_1' &= q_1\,, & p_1' &= p_1 + 2\,\beta_{74}\, q_1\, p_2\, p_3\,,\\ q_2' &= q_2 - \beta_{74}\, q_1^2\, p_3\,, & p_2' &= p_2\,,\\ q_3' &= q_3 - \beta_{74}\, q_1^2\, p_2\,, & p_3' &= p_3\,. \end{aligned}$$

Thus, $e^{:h_7:}$ is a solvable map. The action of the solvable map $e^{:h_8:}$ is similar to $e^{:h_7:}$.

The action of $e^{:h_9:}$ on the phase space variables is obtained as follows. The appropriate Hamiltonian is

$$h_9 = q_1^2\, p_1\, [\beta_2 + \beta_{64}\, q_2 + \beta_{65}\, p_2 + \beta_{66}\, q_3 + \beta_{67}\, p_3] = E_9\,. \tag{17}$$

Here E_9 denotes the conserved numerical value of h_9 and is obtained by evaluating h_9 at $t = 0$. We note that

$$\frac{d}{dt}(q_1^2 p_1) = 2 q_1 p_1 \dot{q}_1 + q_1^2 \dot{p}_1 = 0.$$

Therefore, $q_1^2\, p_1 = q_{1_0}^2\, p_{1_0} = \text{constant}$. Consequently,

$$\beta_2 + \beta_{64}\, q_2 + \beta_{65}\, p_2 + \beta_{66}\, q_3 + \beta_{67}\, p_3 = \frac{E_9}{q_{1_0}^2\, p_{1_0}} = \text{constant}\,.$$

Hamilton's equations of motion, give

$$\dot{q}_1 = \frac{\partial h_9}{\partial p_1} = q_1^2 \left(\beta_2 + \beta_{64}\, q_2 + \beta_{65}\, p_2 + \beta_{66}\, q_3 + \beta_{67}\, p_3\right) .$$

Integrating the above equation, we have

$$q_1 = \frac{q_{1_0}^2\, p_{1_0}}{q_{1_0}\, p_{1_0} - E_9\, t} ,$$

where we have denoted the values of q_i and p_i at time $t = 0$ by q_{i_0} and p_{i_0}. Also, we will sometimes collectively denote q_1, q_2, q_3 by q and p_1, p_2, p_3 by p. Further, if $q_{1_0}, p_{1_0} \neq 0$, we have$q_1 \neq 0$ and from Eq. (17)

$$\begin{aligned} p_1 &= \frac{E_9}{q_1^2 \left(\beta_2 + \beta_{64}\, q_2 + \beta_{65}\, p_2 + \beta_{66}\, q_3 + \beta_{67}\, p_3\right)} , \\ &= p_{1_0} \left(\frac{q_{1_0}}{q_1}\right)^2 . \end{aligned}$$

If $q_{1_0} = 0$, then $q_1 = q_{1_0}$ and $p_1 = p_{1_0}$.

The equation for q_2 is given by

$$\dot{q}_2 = \frac{\partial h_9}{\partial p_2} = \beta_{65}\, q_{1_0}^2\, p_{1_0} .$$

Once again integrating the above equation, we have

$$q_2 = q_{2_0} + \beta_{65}\, q_{1_0}^2\, p_{1_0} t .$$

Similarly, integrating the equations of motion for the other variables and letting $(q,\ p) \to (q',\ p')\,,\ (q_0,\ p_0) \to (q,\ p)\,,\ t \to -1$, we get the action of $e^{:h_9:}$ on the phase space variables. Thus,

$$q_1' = \begin{cases} \dfrac{q_1^2\, p_1}{q_1\, p_1 + E_9} & \text{if} \quad q_1, p_1 \neq 0 \\ q_1 & \text{otherwise} , \end{cases}$$

$$p_1' = \begin{cases} p_1 \left(\dfrac{q_1}{q_1'}\right)^2 & \text{if} \quad q_1, p_1 \neq 0 \\ p_1 & \text{otherwise} , \end{cases}$$

$$\begin{aligned} q_2' &= q_2 - \beta_{65}\, q_1^2\, p_1 , \\ p_2' &= p_2 + \beta_{64}\, q_1^2\, p_1 , \\ q_3' &= q_3 - \beta_{67}\, q_1^2\, p_1 , \\ p_3' &= p_3 + \beta_{66}\, q_1^2\, p_1 . \end{aligned}$$

This shows that $e^{:h_9:}$ is a solvable map. Similarly, $e^{:h_{10}:}$, $e^{:h_{11}:}$, ... , $e^{:h_{14}:}$ can be shown to be solvable maps and their action on phase space variables can be got by appropriately changing q and p in the above equations.

The action of the map $e^{:h_{15}:}$ on the phase space variables can be obtained in the following way. The corresponding Hamiltonian is

$$h_{15} = \beta_{58}\, q_1^3\, p_1 + \beta_{149}\, q_2^3\, p_2 + \beta_{179}\, q_3^3\, p_3\,. \tag{18}$$

From the equations of motion,

$$\dot{q}_1 = \frac{\partial h_{15}}{\partial p_1} = \beta_{58}\, q_1^3\,.$$

Integrating, we have

$$q_1 = \frac{q_{1_0}}{\sqrt{1 - 2\,\beta_{58}\, q_{1_0}^2\, t}}\,. \tag{19}$$

Substituting (19) in the equation of motion for p_1 and integrating, we see that

$$\dot{p}_1 = -3\,\beta_{58}\, q_1^2\, p_1\,, \quad \Rightarrow \quad p_1 = p_{1_0}\, \left(1 - 2\,\beta_{58}\, q_{1_0}^2\, t\right)^{3/2}\,. \tag{20}$$

Similarly integrating the remaining equations of motion and letting $(q,\ p) \to (q',\ p')$, $(q_0,\ p_0) \to (q,\ p)$, $t \to -1$, we get the action of $e^{:h_{15}:}$ on the phase space variables:

$$\begin{aligned}
q_1' &= \frac{q_1}{\sqrt{1 + 2\,\beta_{58}\, q_1^2}}\,, \quad p_1' = p_1\, \left(1 + 2\,\beta_{58}\, q_1^2\right)^{3/2}\,. \\
q_2' &= \frac{q_2}{\sqrt{1 + 2\,\beta_{149}\, q_2^2}}\,, \quad p_2' = p_2\, \left(1 + 2\,\beta_{149}\, q_2^2\right)^{3/2}\,. \\
q_3' &= \frac{q_3}{\sqrt{1 + 2\,\beta_{179}\, q_3^2}}\,, \quad p_3' = p_3\, \left(1 + 2\,\beta_{179}\, q_3^2\right)^{3/2}\,.
\end{aligned}$$

Thus, $e^{:h_{15}:}$ is a solvable map. The action of the map $e^{:h_{16}:}$ is similar to that of $e^{:h_{15}:}$ and can be got by interchanging q and p and by letting $t \to -t$.

Finally, the action of the solvable map $e^{:h_{20}:}$ is as follows:

$$\begin{aligned}
q_1' &= q_1 \exp\left[-\left(2\,\beta_{63}\, q_1\, p_1 + \beta_{84}\, q_2\, p_2 + \beta_{91}\, q_3\, p_3\right)\right], \\
p_1' &= p_1 \exp\left[\left(2\,\beta_{63}\, q_1\, p_1 + \beta_{84}\, q_2\, p_2 + \beta_{91}\, q_3\, p_3\right)\right], \\
q_2' &= q_2 \exp\left[-\left(\beta_{84}\, q_1\, p_1 + 2\,\beta_{152}\, q_2\, p_2 + \beta_{162}\, q_3\, p_3\right)\right], \\
p_2' &= p_2 \exp\left[\left(\beta_{84}\, q_1\, p_1 + 2\,\beta_{152}\, q_2\, p_2 + \beta_{162}\, q_3\, p_3\right)\right], \\
q_3' &= q_3 \exp\left[-\left(\beta_{91}\, q_1\, p_1 + \beta_{162}\, q_2\, p_2 + 2\,\beta_{182}\, q_3\, p_3\right)\right], \\
p_3' &= p_3 \exp\left[\left(\beta_{91}\, q_1\, p_1 + \beta_{162}\, q_2\, p_2 + 2\,\beta_{182}\, q_3\, p_3\right)\right].
\end{aligned}$$

Therefore $e^{:h_{20}:}$ is also a solvable map.

Using the above solvable map method, different nonlinear Hamiltonian systems were studied and good results were obtained. Since the results obtained are very similar to those already demonstrated in Ref. [12] for lower dimensions, the details are omitted here.

4 Conclusions

We have studied a symplectic integration algorithm using solvable maps. The solvable map factorization for three degrees of freedom (up to order 4) was carried out and it involves 20 solvable maps. It has been shown [12] that this algorithm gives good results for various examples. It also provides a fast and accurate symplectic integration method for complicated Hamiltonian systems. Another advantage of this method is that it can be easily extended to higher dimensions since the computations involved are not that difficult when symbolic manipulation programs are used.

Acknowledgements

This work was supported by grants from DRDO and ISRO, India as part of funding for Nonlinear Studies Group. GR is also associated with the Centre for Theoretical Studies, Indian Institute of Science and is a honorary faculty member of the Jawaharlal Nehru Center for Advanced Scientific Research, Bangalore.

References

1. Sanz-Serna, J. M. and Calvo, M. P.: Numerical Hamiltonian Problems. Chapman & Hall, London (1994) and references therein
2. Ruth, R. D.: A canonical integration technique. IEEE Trans. Nucl. Sci. **30** (1983) 2669–2671
3. Irwin, J.: A multi-kick factorization algorithm for nonlinear symplectic maps. SSC Report No. 228 (1989)
4. Rangarajan, G.: Invariants for symplectic maps and symplectic completion of symplectic jets. Ph.D Thesis. University of Maryland (1990)
5. Channell, P. J. and Scovel, C.: Symplectic integration of Hamiltonian systems. Nonlinearity **3** (1990) 231–259
6. Yoshida, H.: Construction of higher order symplectic integrators. Phys. Lett. A **150** (1990) 262–268
7. Rangarajan, G.: Symplectic completion of symplectic jets. J. Math. Phys. **37** (1996) 4514–4542. For associated group theoretical material, see Rangarajan, G.: Representations of Sp(6,R) and SU(3) carried by homogeneous polynomials. J. Math. Phys. **38** (1997) 2710–2719
8. Forest, E. and Ruth, R.: Fourth order symplectic integration. Physica D **43** (1990) 105–117
9. Dragt, A. J. and Abell, D. T.: Jolt factorization of symplectic maps. Int. J. Mod. Phys. **2B** (1993) 1019
10. Rangarajan, G.: Jolt factorization of the pendulum map. J. Phys. A: Math. Gen. **31** (1998) 3649–3658
11. Rangarajan, G., Dragt, A. J. and Neri, F.: Solvable map representation of a nonlinear symplectic map. Part. Accel. **28** (1990) 119–124
12. Rangarajan, G. and Sachidanand, M.: Symplectic integration using solvable maps. J. Phys. A: Math. Gen. **33** (2000) 131–142

13. Dragt, A. J.: Lectures on nonlinear orbit dynamics. In: Carrigan, R. A., Huson, F. and Month, M. (eds.): Physics of High Energy Particle Accelerators. American Institute of Physics, New York (1982) 147–313; Dragt, A. J., Neri, F., Rangarajan, G., Douglas, D. R., Healy, L. M. and Ryne, R. D.: Lie algebraic treatment of linear and nonlinear beam dynamics. Annu. Rev. Nucl. Part. Sci. **38** (1988) 455–496
14. Sachidanand, M.: Applications of Lie Algebraic Techniques to Hamiltonian Systems. Ph. D. Thesis. Indian Institute of Science (2000)
15. Cornwell, J. F.: Group Theory in Physics. Volume 1. Academic, London (1984)

A Parallel ADI Method for a Nonlinear Equation Describing Gravitational Flow of Ground Water

I.V. Schevtschenko

Rostov State University
Laboratory of Computational Experiments on Super Computers
34/1 Communistichesky avenue, Apt. 111
344091, Rostov-on-Don, Russia
ishevtch@uic.rnd.runnet.ru

Abstract. The aim of the paper is an elaboration of a parallel alternating-direction implicit, or ADI, method for solving a non-linear equation describing gravitational flow of ground water and its realization on a distributed-memory MIMD-computer under the MPI message-passing system. Aside from that, the paper represents an evaluation of the parallel algorithm in terms of relative efficiency and speedup. The obtained results show that for reasonably large discretization grids the parallel ADI method is effective enough on a large number of processors.

Keywords: gravitational flow of ground water, finite difference method, Peaceman-Rachford difference scheme, parallel ADI method, conjugate gradient method.

1 Introduction

The past decades a substantial progress has been made in mathematical description of water flow and pollutant transport processes. Many today's mathematical models are available to predict admixtures migration in ground water under diverse conditions of processes progress. Approximation of such models generates large systems of linear algebraic or differential equations that demands utilizing modern supercomputers proposing powerful computational resources to solve large problems in various fields of science. In particular, by solving the equation of pollutant transport in ground water it is necessary to know a level of ground water in a water-bearing stratum described by the balance mass equation. In this paper we consider a parallel solution of that equation for approximation of which the finite difference method is used. The solution of the problem is found with the aid of the ADI method, in particular, with using Peaceman-Rachford difference scheme [8]. We exploit natural parallelism of the difference scheme to apply it to distributed-memory MIMD-computers.

An outline of the paper is as follows. Section 2 introduces to general formulation and numerical approximation of the original mathematical model, represents

V.N. Alexandrov et al. (Eds.): ICCS 2001, LNCS 2073, pp. 904–910, 2001.
© Springer-Verlag Berlin Heidelberg 2001

some results on accuracy and stability of the used difference scheme and substantiates applicability of the conjugate gradient (CG) method to solving systems of linear algebraic equations (SLAEs) generated by Peaceman-Rachford difference scheme. Section 3 is represented by a parallel realization of the ADI method. In the same place we evaluate the parallel algorithm in terms of relative efficiency and speedup. Finally, in section 4, we give our conclusions.

2 General Formulation and Numerical Approximation

One of the existing non-linear models describing gravitational flow of ground water in an anisotropic element of a water-bearing stratum Ω can be represented in the form [2]

$$a^* \frac{\partial h}{\partial t} = \sum_{i=1}^{2} \frac{\partial}{\partial x_i} \left(\rho h k \frac{\partial h}{\partial x_i} \right) + \rho(v_{(x)} + v_{(y)}). \tag{1}$$

Here $x = x_1$, $y = x_2$; ρ is the water density, $h(x, y, t)$ is the level of ground water, k is the filtrational coefficient, $v_{(x)}$ and $v_{(y)}$ are the filtration velocities from below and from above of the water-bearing stratum respectively. Parameter $a^* > 0$ depends on physical characteristics of the water-bearing stratum.

For equation (1) we can define the initial condition

$$h(x, y, t = 0) = h_0(x, y),$$

where $h_0(x, y)$ is a given function and Dirichlet boundary value problem

$$h|_{\partial\Omega} = f(x, y).$$

Here $f(x, y)$ is a function prescribed on the boundary of the concerned field.

Approximation of equation (1) bases on Peaceman-Rachford difference scheme, where along with the prescribed grid functions $h(x, y, t)$ and $h(x, y, t+\tau)$ an intermediate function $h(x, y, t + \frac{\tau}{2})$ is introduced. Thus, passing (from n time layer to $n + 1$ time layer) is performed in two stages with steps 0.5τ, where τ is a time step.

Let us introduce a grid of a size $M \times N$ in a simply connected area $\Omega = [0, a] \times [0, b]$ with nodes $x_i = i\Delta x$, $y_j = j\Delta y$, where $i = 1, 2, \ldots M$, $j = 1, 2, \ldots N$, $\Delta x = \frac{a}{M}$, $\Delta y = \frac{b}{N}$. By denoting $h = h^n$, $\bar{h} = h^{n+\frac{1}{2}}$, $\hat{h} = h^{n+1}$, $h_x = \frac{h_{i+1,j} - h_{ij}}{2}$, $h_y = \frac{h_{i,j+1} - h_{ij}}{2}$, $h_{\bar{x}} = \frac{h_{ij} - h_{i-1,j}}{2}$, $h_{\bar{y}} = \frac{h_{ij} - h_{i,j-1}}{2}$, $h_{\underline{x}} = \frac{h_{i+1,j} + h_{ij}}{2}$, $h_{\underline{y}} = \frac{h_{i,j+1} + h_{ij}}{2}$, $h_{\tilde{x}} = \frac{h_{i-1,j} + h_{ij}}{2}$, $h_{\tilde{y}} = \frac{h_{i,j-1} + h_{ij}}{2}$ let us write out the difference approximation for equation (1)

$$\begin{cases} a^* \frac{\bar{h}_{ij} - h_{ij}}{0.5\tau} = \frac{\rho_{\underline{x}} k_{\underline{x}} h_{\underline{x}} \bar{h}_x - \rho_{\tilde{x}} k_{\tilde{x}} h_{\tilde{x}} \bar{h}_{\tilde{x}}}{\Delta x} + \frac{\rho_{\underline{y}} k_{\underline{y}} h_{\underline{y}} h_y - \rho_{\tilde{y}} k_{\tilde{y}} h_{\tilde{y}} h_{\bar{y}}}{\Delta y} + \phi, \\ a^* \frac{\hat{h}_{ij} - \bar{h}_{ij}}{0.5\tau} = \frac{\rho_{\underline{x}} k_{\underline{x}} h_{\underline{x}} \bar{h}_x - \rho_{\tilde{x}} k_{\tilde{x}} h_{\tilde{x}} \bar{h}_{\tilde{x}}}{\Delta x} + \frac{\rho_{\underline{y}} k_{\underline{y}} \bar{h}_{\underline{y}} \hat{h}_y - \rho_{\tilde{y}} k_{\tilde{y}} \bar{h}_{\tilde{y}} \hat{h}_{\bar{y}}}{\Delta y} + \phi. \end{cases} \tag{2}$$

Here $\phi = \rho_{ij}(v_{(x)ij} + v_{(y)ij})$. The difference approximation of the initial condition and Dirichlet boundary value problem can be represented as

$$h|_{t=0} = h_{(0)ij}, \quad h|_{\partial\Omega} = f_{ij}. \tag{3}$$

By addressing stability investigation of equation (2) we formulate the following lemma

Lemma 1 *Peaceman-Rachford difference scheme for equation (1) with Dirichlet's boundary conditions at $a^* > 0$ is stable.*

Concerning the difference scheme (2) it can be noted that it has second approximation order [9] both in time and in space.

By using natural regulating of unknown values in the computed field let us reduce difference problem (2), (3) to the necessity of solving SLAEs $A_k u_k = f_k, k = 1, 2$ with special matrices. The coefficient matrices $A_k, k = 1, 2$ are not constant here. The obtained SLAEs had been scaled, i.e. the elements of the coefficient matrices and RHSs: $A_k = (a_{ij})_k^{MN}, f_k, k = 1, 2$ had the following form

$$\hat{a}_{ij} = \frac{a_{ij}}{\sqrt{a_{ii}a_{jj}}}, \ \hat{f}_i = \frac{f_i}{a_{ii}}, \ i, j = 1, 2, \ldots, MN$$

and solved with the CG method [9] afterwards. The selection of the CG method is based on its acceptable calculation time in comparison with simple iteration, Seidel, minimal residual and steepest descent methods [5].

To proceed, we note that from previous lemma we can infer the appropriateness of using the CG method since

$$A_k = (A_k)^T, \ A_k > 0, \ k = 1, 2.$$

3 Algorithm Parallel Scheme

Before passage to the description of the parallel algorithm we would like to say a few words about the computational platform on which the algorithm has been run and the library with the help of which it has been realized.

The computational system nCube 2S is a MIMD-computer of hypercubic architecture. The number of computational nodes is 2^n, $n \leq 13$. These nodes are unified into communication scheme of a multidimensional cube with maximum length of a communication line equaled to n. Such a communication scheme allows to transmit messages fast enough (channel capacity is 2.5 MBytes/s) irrespective of computational process since each node has a communication coprocessor aside from a computational processor. At our disposal we had the described system in reduced version: 64 nodes with peak performance of 128 MFlops and 2048 MBytes of memory.

Relative to the paradigm of message passing it can be noted that it is used widely on certain classes of parallel machines, especially those with distributed

memory. One of representatives of this conception is MPI (Message Passing Interface) [6]. As we can see from the title, MPI destines for supporting parallel applications to work in terms of the message passing system and allows to use its functions in C/C++ and Fortran 77/90. Besides, amongst a number of books devoted to various aspects of using MPI we can mention, for instance, the following [3], [4], [7].

The parallel algorithm for solving equation (2), as mentioned above, bases on natural parallelism which is suggested by Peaceman-Rachford difference scheme. Using the ADI method gives an opportunity to exploit any method to solve SLAEs obtained on $n + \frac{1}{2}$ and $n + 1$ time layers. In our case we use the CG method. Along with it, application of Peaceman-Rachford difference scheme to equation (1) allows to find numerical solution of the SLAEs on each time layer independently, i.e. irrespective of communication process. The main communication loading lies on connection between two time layers. Thus, one step of the algorithm to be executed requires two interchanges of data at passage to $n + \frac{1}{2}$ and $n + 1$ time layers.

As mentioned before, the ADI method generates two SLAEs with special matrices. One of those matrices obtained on $n+\frac{1}{2}$ time layer is a band tridiagonal matrix and consequently can be transformed by means of permutation of rows to a block tridiagonal matrix, while the second one, obtained on $n+1$ time layer, is a block tridiagonal matrix primordially.

Taking into account aforesaid one step of the parallel algorithm for solving equation (2) with SLAEs $A_k X_k = B_k,\ k = 1, 2$ can be represented in the following manner

1. Compute B_1 on $n + \frac{1}{2}$ time layer.
2. Make the permutation of vectors $X_1^{(0)}$, B_1, where $X_1^{(0)}$ is an initial guess of the CG method on $n + \frac{1}{2}$ time layer.
3. Solve equation $A_1 X_1 = B_1$ on $n + \frac{1}{2}$ time layer with the CG method.
4. Compute B_2 on $n + 1$ time layer step partially, i.e. without the last item of the second equation (2).
5. Make the permutation of vectors $X_2^{(0)} = X_1$, B_2, where $X_2^{(0)}$ is an initial guess of the CG method on $n + 1$ time layer.
6. Compute the missing item so as the computation of B_2 on $n + 1$ time layer has been completed.
7. Solve equation $A_2 X_2 = B_2$ on $n + 1$ time layer with the CG method.
8. Set $X_1^{(0)} = X_2$ and go to point 1 to do the next step of the algorithm.

Let us consider the described algorithm in more detail. Suppose, we have p processors and it is to solve a system of a size $M \times N$. We proceed from the assumption that $\left\{\frac{M}{p}\right\} = 0$ and $\left\{\frac{N}{p}\right\} = 0$, where $\{x\}$ is a fractional part of number x, i.e. vectors $X_k^{(0)},\ B_k,\ k = 1, 2$ are distributed uniformly.

First step of the algorithm is well-understood while the second one claims more attention. Let vectors $X_1^{(0)}$, B_1 be matrices (which are distributed in the rowwise manner) consist of elements of corresponding vectors, then to solve

equation $A_1X_1 = B_1$ on $n + \frac{1}{2}$ time layer with the CG method in parallel we need to transpose the matrix corresponding to vector $B_1 = \{b_1, b_2, \ldots, b_{MN}\}$

$$\begin{pmatrix} b_1 & b_2 & \ldots & b_N \\ b_{N+1} & b_{N+2} & \ldots & b_{2N} \\ \ldots & \ldots & \ldots & \ldots \\ b_{(M-1)N+1} & b_{(M-1)N+2} & \ldots & b_{MN} \end{pmatrix} \to \begin{pmatrix} b_1 & b_{N+1} & \ldots & b_{(M-1)N+1} \\ b_2 & b_{N+2} & \ldots & b_{(M-1)N+1} \\ \ldots & \ldots & \ldots & \ldots \\ b_N & b_{2N} & \ldots & b_{MN} \end{pmatrix}$$

and the matrix corresponding to vector $X_1^{(0)}$. Of course, such a transposition requires transmission of some sub-matrices ($\frac{M}{p} \times \frac{N}{p}$ size) to the corresponding processors. Thus, the number of send/receive operations $C_{s/r}$ and the amount of transmitted data C_t (in element equivalent) are

$$C_{s/r} = 2p(p-1), \quad C_t = 2M\left(N - \frac{N}{p}\right).$$

Further, in accordance with the algorithm to avoid extra communications we compute vector B_2 partially and then permute vectors $X_2^{(0)} = X_1$, B_2 as above. Afterwards, we complete the computation of vector B_2 (its missing item) and solve equation $A_2X_2 = B_2$ on $n+1$ time layer with the CG method in parallel. By resuming aforesaid one step of the algorithm to be run requires

$$C_{sr} = 4p(p-1), \quad C_t = 4M\left(N - \frac{N}{p}\right),$$

$$C_c = \frac{N}{p}\left((25m-7)\left(I_{CG}^{n+\frac{1}{2}} + I_{CG}^{n+1}\right) + 90M - \frac{8M}{p} - 26\right) + 24M + 4.$$

Here $I_{CG}^{n+\frac{1}{2}}$ and I_{CG}^{n+1} are the number of iterations of the CG method in solving equation (2) on $n+\frac{1}{2}$ and $n+1$ time layers, and C_c is a computational complexity of the algorithm.

At this we finish the description of the parallel algorithm and consider some test experiments all of which are given for one step of the algorithm and for $I_{CG}^{n+\frac{1}{2}} = I_{CG}^{n+1} = 1$. The horizontal axis, in all the pictures, is $2^p, p = 0, 1, \ldots, 6$.

By following [10] let us consider relative efficiency and speedup

$$S_p = \frac{T_1}{T_p}, \quad E_p = \frac{S_p}{p},$$

where T_p is a time to run a parallel algorithm on a computer with p processors ($p > 1$), T_1 is a time to run a sequential algorithm on one processor of the same computer.

As we can see from figure 1 relative speedup and efficiency are satisfactory even for a grid of $N = M = 512$ size.

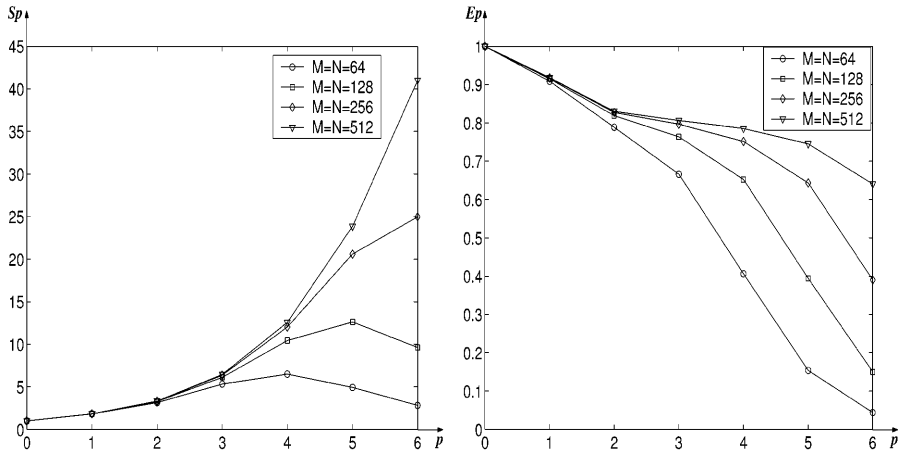


Fig. 1. Relative speedup (to the left) and efficiency (to the right) of the algorithm at various grid sizes.

4 Conclusion

In conclusion we would like to say a few words about further work which will be aimed at elaboration of a parallel ADI method for solving the following equation

$$a^* \frac{\partial h}{\partial t} = \sum_{i=1}^{2} \frac{\partial}{\partial x_i} \left(\rho h k \frac{\partial h}{\partial x_i} \right) + \sum_{i=1}^{2} \frac{\partial}{\partial x_i} \left(\rho h k \frac{\partial \zeta}{\partial x_i} \right) + \rho \left(v_{(x)} + v_{(y)} \right),$$

which is one of the base equations in solving the problem of gravitational flow of ground water.

References

1. R. Barrett, M. Berry, T.F. Chan, J. Demmel, J.M. Donato, J. Dongarra, V. Eijkhout, R. Pozo, C. Romine, Henk Van der Vorst, Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods, http://www.netlib.org/templates/Templates.html
2. J. Bear, D. Zaslavsky, S. Irmay, Physical principles of water percolation and seepage. UNESCO, (1968)
3. I. Foster, Designing and Building Parallel Programs: Concepts and Tools for Parallel Software Engineering, Addison-Wesley Pub. Co., (1995)
4. W. Gropp, E. Lusk, A. Skjellum, T. Rajeev, Using MPI: Portable Parallel Programming With the Message-Passing Interface, Mit press, (1999)
5. L.A. Krukier, I.V. Schevtschenko, Modeling Gravitational Flow of Subterranean Water. Proceedings of the Eighth All-Russian Conference on Modern Problems of Mathematical Modeling, Durso, Russia, September 6-12, RSU Press, (1999), 125-130

6. *MPI: A Message-Passing Interface Standard, Message Passing Interface Forum*, (1994)
7. P. Pacheco, Parallel Programming With MPI, Morgan Kaufamnn Publishers, (1996)
8. D. Peaceman and J. H.H. Rachford, The numerical solution of parabolic and elliptic differential equations. J. Soc. Indust. Appl. Math., No.3 (1955), 28-41
9. A.A. Samarskii and A.V. Goolin, Numerical Methods, Main Editorial Bord for Physical and Mathematical Literature, (1989)
10. Y. Wallach, Alternating Sequential/Parallel Processing, Springer-Verlag, (1982)

The Effect of the Cusp on the Rate of Convergence of the Rayleigh-Ritz Method

Ioana Sîrbu
Department of Mathematics
244 Mathematics Building
SUNY at Buffalo
Buffalo, NY 14260-2900

Harry F. King
Department of Chemistry
330 Natural Science Complex
SUNY at Buffalo
Buffalo, NY 14260

Abstract. This paper investigates how smoothing the Hamiltonian and the cusp of the corresponding eigenfunction affects the rate of convergence of the Rayleigh-Ritz method. A simple example from quantum mechanics is used, with a basis of harmonic oscillator functions.

1 Introduction

This study is motivated by a computational problem in the electronic molecular structure theory. It has been shown ([1]) that the variational energy error of a configuration interaction(CI), or any other orbital-based method is slow (of order $O(L^{-3})$ or greater, where L is the maximum angular momentum in the finite orbital basis). This behavior can be explained by the inability of the basis functions to describe the "electron correlation cusps" of the wavefunction introduced by the singularities of the Coulombic potential.

One possible approach that we are exploring is a perturbational one, in which the reference problem has a Hamiltonian free of such singularities, and for which the wavefunctions differ significantly from those of the true Hamiltonian only in the vicinity of such cusps. Traditional CI methods are used to solve the reference problem, and geminal-based methods are employed to solve the low-order Rayleigh-Schrödinger perturbation equations. The success of this approach is dependent upon finding a reference Hamiltonian for which the Rayleigh-Ritz (RR) method converges far more rapidly than for the true Hamiltonian (see [2]).

This paper illustrates how the convergence of the RR method is accelerating with the "smoothing" of the Hamiltonian and of the corresponding ground-state wavefunction for a simple example from quantum mechanics. The singular potential v used here is different from the usual potentials used in quantum chemistry, so we investigate whether the associated operator H is selfadjoint and the RR method for this operator using a basis of harmonic oscillator functions is convergent.

V.N. Alexandrov et al. (Eds.): ICCS 2001, LNCS 2073, pp. 911–917, 2001.
© Springer-Verlag Berlin Heidelberg 2001

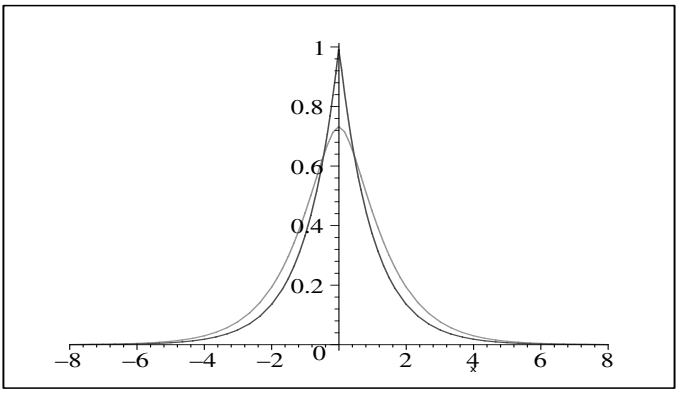


Fig. 1. The wavefunctions ψ and ψ_a for $a = 1$

2 The Model

Consider the one-dimensional Schrödinger equation

$$-\frac{1}{2}\frac{d^2\psi(x)}{dx^2} + v(x)\psi(x) = E\psi(x),\ x \in \mathbf{R} \tag{1}$$

With the potential $v(x) = -\delta(x)$ the equation (1) has the ground-state energy $E = -1/2$, with the normalized wavefunction $\psi(x) = exp(-|x|)$. With the smoothed potential

$$v_a(x) = -\frac{a + 1/2 + a(a+2)|x| + a^2x^2}{(1 + a|x|)^4} \tag{2}$$

the Schrödinger equation has ground-state wavefunction $\psi_a(x) = N_a exp\left(-\frac{ax^2}{1+a|x|}\right)$ with the same energy $E = -1/2$; N_a is the normalization constant. The function $\psi(x)$ is continuous, but has a cusp at the origin; the function $\psi_a(x)$ has continuous first and second derivatives and a discontinuous third derivative for any $a > 0$. Moreover, $\psi_a(x) \to \psi(x)$ pointwise as $a \to \infty$. $v_a(x) \to 0$ as $a \to \infty$ for any $x \neq 0$ and$v_a(0) = -a - 1/2 \to -\infty$ as $a \to \infty$.

3 The Expansion

Consider the expansion of the two wavefunctions, ψ and ψ_a

$$\psi(x) = \sum_{n=0}^{\infty} c_n\phi_n(x), \qquad \psi_a(x) = \sum_{n=0}^{\infty} c_n^a\phi_n(x) \tag{3}$$

in the orthonormal basis of harmonic oscillator functions

$$\phi_m(x) = N_m H_m(\lambda x) exp(-(\lambda x)^2/2), \tag{4}$$

where λ is a positive scaling factor and N_m is a normalization constant.

Since both ψ and ψ_a are even functions, $c_{2k+1} = c^a_{2k+1} = 0$ for all integers k. For even integers we have $c_{2k} = C_{2k}$ and $c^a_{2k} = C^a_{2k}$, where

$$C_n = \frac{2N_n}{\lambda} \int_0^\infty exp\left(-\frac{x}{\lambda} - \frac{x^2}{2}\right) H_n(x)dx \tag{5}$$

$$C^a_n = \frac{2N_n N_a}{\lambda} \int_0^\infty exp\left(-\frac{x^2}{2} - \frac{ax^2}{\lambda^2 + a\lambda x}\right) H_n(x)dx \tag{6}$$

The coefficients C_n can be computed using an exact recurrence formula which can be obtained integrating (5) by parts. To compute C^a_n an approximate quadrature formula is used for the function $f(x) = exp\left(-\frac{x^2}{2} - \frac{ax^2}{\lambda^2+\lambda ax}\right) H_n(x)$, where the roots x_j and weights w_j are for polynomials orthogonal with respect to the weight function $w(x) = exp(-x^2)$ on the interval $(0, \infty)$.

Once the expansion coefficients c_n and c^a_n are computed, one can define the projection of the wavefunctions ψ and ψ_a on the $n+1$-dimensional space W_n spanned by $\phi_0, \phi_1, \ldots, \phi_n$:

$$\psi_n(x) = \sum_{i=0}^{n} c_i \phi_i(x), \qquad \psi^a_n(x) = \sum_{i=0}^{n} c^a_i \phi_i(x). \tag{7}$$

$$H = -\frac{1}{2}\frac{d^2}{dx^2} + v(x), \qquad H^a = -\frac{1}{2}\frac{d^2}{dx^2} + v_a(x). \tag{8}$$

Next, let us define E_n and E^a_n as the lowest eigenvalue of the Hamiltonian matrix $(< \phi_i|H|\phi_j >)_{0\le i,j\le n}$ and of $(< \phi_i|H^a|\phi_j >)_{0\le i,j\le n}$, respectively. For each n, the norm of the projection $\|\psi_n\|$ is maximized as a function of λ and E_n is computed for this λ. Values of E_n are reported in Table 1 and E^a_n are reported in Table 2 for two different values of the smoothing parameter a. Also

Table 1. λ and the energies E^1_n and E_n for $\psi(x)$

ψ				
n	λ	(ψ_n, ψ_n)	E^1_n	E_n
8	1.300	.997350	-.405894	-.409083
16	1.450	.999183	-.433470	-.435312
24	1.575	.999629	-.447763	-.448960
32	1.675	.999795	-.456453	-.457308
40	1.750	.999873	-.462174	-.462830
64	1.975	.999955	-.472857	-.473204
80	2.125	.999973	-.477236	-.477483
104	2.275	.999986	-.481202	-.481372
120	2.375	.999990	-.483174	-.483311

Table 2. λ and the energies $E_n^{1,a}$ and E_n^1 for $\psi_a(x)$for $a = 4$ and $a = 1$

	ψ_a for $a = 4$				ψ_a for $a = 1$			
n	λ	(ψ_n^a, ψ_n^a)	$E_n^{1,a}$	E_n^a	λ	(ψ_n^a, ψ_n^a)	$E_n^{1,a}$	E_n^a
8	1.125	.999051	-.485252	-.485353	.875	.999811	-.498480	-.498483
16	1.225	.999824	-.494604	-.494620	.950	.999979	-.499683	-.499684
24	1.300	.999947	-.497446	-.497450	1.00	.999995	-.499891	-.499891
32	1.375	.999979	-.498464	-.498467	1.05	.999998	-.499975	-.499975
40	1.400	.999989	-.498934	-.498938	1.075	.999999	-.499975	-.499997
64	1.550	.999998	-.499752	-.499752				
80	1.625	.999999	-.499869	-.499869				

reported in these tables are energies E_n^1 and $E_n^{1,a}$ computed for ψ_n and ψ_n^a, the projections of the true wavefunctions on the subspace.

$$E_n^1 = \frac{<\psi_n|H|\psi_n>}{(\psi_n, \psi_n)}, \qquad E_n^{1,a} = \frac{<\psi_n^a|H^a|\psi_n^a>}{(\psi_n^a, \psi_n^a)}. \tag{9}$$

Note that $E_n \leq E_n^1$ and $E_n^a \leq E_n^{1,a}$. Figures 2 and 3 illustrate the beneficial effects of smoothing.

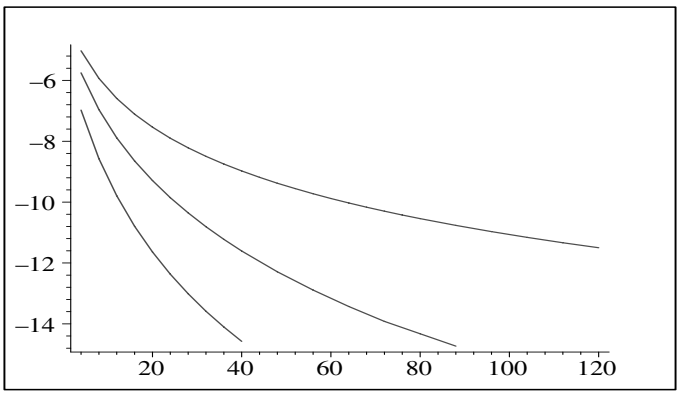


Fig. 2. The norm errors $ln(1 - (\psi_n, \psi_n))$ (top curve) and $ln(1 - (\psi_n^a, \psi_n^a))$ for $a = 1$ (lowest curve) and $a = 4$ (middle curve)

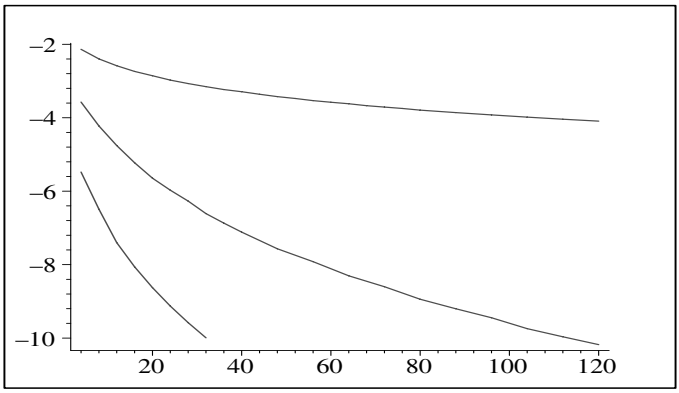


Fig. 3. The energy errors $ln(1/2 + E_n)$ (top curve) and $ln(1/2 + E_n^a)$ for $a = 1$ (lowest curve) and and $a = 4$ (middle curve).

4 The Convergence of the RR Method

For eq. (1) with $v(x) = -\delta(x)$ the behavior of the energy error exhibited in Figure 3 raises a serious question whether the method is just slowly convergent or not convergent at all. In fact, convergence can be established by rigorous mathematics outlined below.

A theorem in Michlin [6] (see also [5]) says that for a positive definite and selfadjoint operator B with the domain $D_B \subset L^2$ dense in L^2 the RR method converges to the lowest exact eigenvalue E_0 of the operator B provided that the basis used $\{\phi_m\}_{m=0,1,2,\ldots}$ is complete in the energy space H_B. The energy space H_B is the closure of D_B in the B-norm:

$$\|f\|_B = (f, Bf)^{1/2}. \tag{10}$$

Let t be the form defined by

$$t(u,v) = \frac{1}{2}\int u^{'}(t)\overline{v^{'}(t)}dt + \left(\frac{1}{2} + \delta_0\right)\int u(t)\overline{v(t)}dt - u(0)\overline{v(0)} \tag{11}$$

for functions $u, v \in H^1$,where $\delta_0 > 0$. The Sobolev space $H^1 = \{f \in L^2, f^{'} \in L^2\}$ (the derivatives are in the generalized sense), is the closure of $C_0^\infty(\mathbf{R})$ in the H^1-norm: $\|f\|_{H^1}^2 = \int |f(t)|^2 dt + \int |f^{'}(t)|^2 dt$. For any $f \in C_0^\infty(\mathbf{R})$ and $\epsilon > 0$

$$|f(0)|^2 \leq \frac{1}{2\epsilon}\int |f(t)|^2 dt + \frac{\epsilon}{2}\int |f^{'}(t)|^2 dt \tag{12}$$

The inequality (12) with $\epsilon = 1$ makes it possible to define $f(0)$ for any $f \in H^1$ and to prove that the symmetric form t is positive definite ($t(u,u) \geq \delta_0\|u\|^2 \; \forall \; u \in$

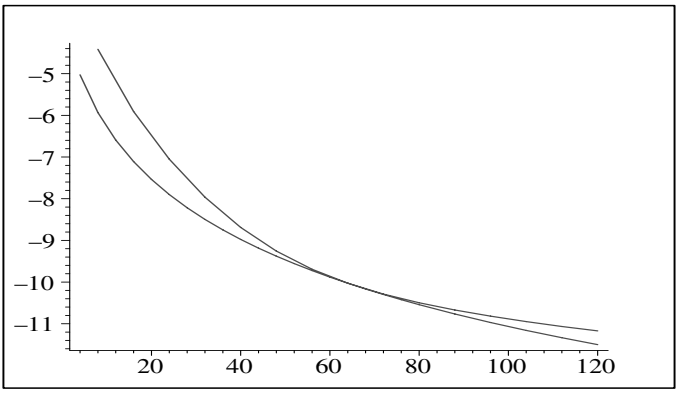


Fig. 4. Norm errors for $\psi(x)$ for $\lambda = 2$(top curve) and optimized λ(lower curve)

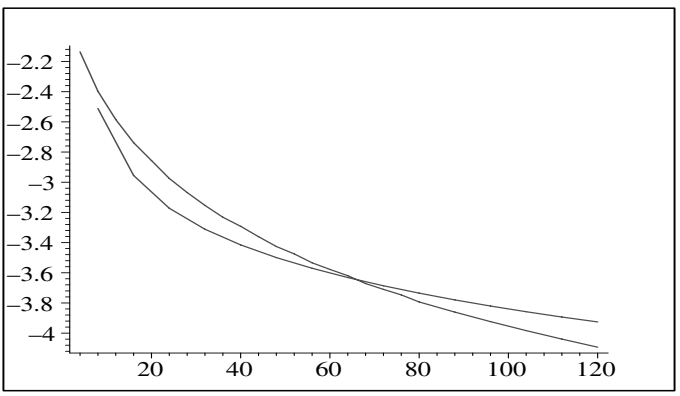


Fig. 5. Energy errors for $\psi(x)$ for xed $\lambda = 2$ (upper right) and optimized λ (lower right)

$H^1,\ \delta_0 > 0$). In its general form, (12) is the main ingredient in proving that the form t is closed. By a representation theorem (see [3]) there exists a selfadjoint operator $T : D(T) \subset H^1 \subset L^2 \to L^2$, positive definite (with the same lower bound δ_0 as the form t) defined by the relation $(Tu, v) = t(u, v)\ \forall u \in D(T),\ \forall v \in H^1$. Moreover, its domain $D(T)$ is dense in the Hilbert space H^1 with the norm $\| \ \|_{H_t} = t(\ ,\ )^{1/2}$.

This implies that H_T-the closure of $D(T)$ in the norm $\| \ \|_{H_t}$-is H^1, since the norms $\| \ \|_{H_t}$ and $\| \ \|_{H^1}$ are equivalent. The basis $\{\phi_m(x)\}_{m=0,1,\ldots}$ is complete in H^1 for any $\lambda > 0$ (see [4]) so the RR method is convergent for the operator T and also for $T - 1/2 - \delta_0$, which is the operator from eq. (1) with $v(x) = -\delta(x)$. These considerations are for a fixed λ, while the results from Figure 2 and 3 are for a λ varied to optimize (ψ_n, ψ_n) for each n. The difference between the results for a fixed λ and λ optimized in the sense above can be seen in Figures 4 and 5. As shown in Table 2, $\lambda = 2$ maximizes the norm of ψ_n for $n \approx 70$, so

the curves in Figures 4 and 5 coincide at this point. The norm error for optimal λ is smaller than that for $\lambda = 2$ for all n, as expected, while the energy error is lower for $\lambda = 2$ for small values of n (λ was optimized with respect to the norm). After $n \approx 70$ the energy error for fixed λ is greater than that for optimized λ. So the method is convergent both for fixed and optimized λ, but the convergence is very slow due to the inability of the basis functions to describe the cusp $\psi(x)$.

References

[1] R. N. Hill *Rates of Convergence and error estimates formulas for the Rayleigh-Ritz variational method*, J. Chem. Phys. **83**, 1173-1196 (1985)

[2] H. F. King *The electron correlation cusp I.Overview and partial wave analysis of the Kais function*, Theoretica Chimica Acta **94**, 345-381 (1996)

[3] T. Kato *Perturbation Theory for Linear Operators*, Springer-Verlag (1980), p.322

[4] T. Kato *Fundamental Properties of Hamiltonian Operators of Schrödinger type*, Trans. Am. Math. Soc., **195**, (1957)

[5] B. Klahn and W. Bingel *The Convergence of the Rayleigh-Ritz Method in Quantum Chemistry I. The Criteria of Convergence*, Theoretica Chimica Acta **47**, 9-26 (1977)

[6] S. G. Michlin *Variationsmethoden der Mathematischen Physik*, Berlin, Akademie Verlag (1962), p.79

The AGEB Algorithm for Solving the Heat Equation in Three Space Dimensions and Its Parallelization Using PVM

Mohd Salleh Sahimi[1], Norma Alias[2], and Elankovan Sundararajan[2]

[1] Department of Engineering Sciences and Mathematics,
Universiti Tenaga Nasional, 43009 Kajang, Malaysia
Sallehs@uniten.edu.my,
[2] Department of Industrial Computing,
Universiti Kebangsaan Malaysia, 43600 UKM, Malaysia
norm_ally@hotmail.com
elan@ftsm.ukm.my

Abstract. In this paper, a new algorithm in the class of the AGE method based on the Brian variant (AGEB) of the ADI is developed to solve the heat equation in 3 space dimensions. The method is iterative, convergent, stable and second order accurate with respect to space and time. It is inherently explicit and is therefore well suited for parallel implementation on the PVM where data decomposition is run asynchronously and concurrently at every time level. Its performance is assessed in terms of speed-up, efficiency and effectiveness.

1 Introduction

The ADI method deals with two-dimensional parabolic (and elliptic) problems. Since the method has no analogue for the one-dimensional case, in [1] the alternating group explicit method which offers its users many advantages was developed. It is shown to be extremely powerful and flexible. It employs the fractional splitting strategy of Yanenko [2] which is applied alternately at each intermediate time step on tridiagonal systems of difference schemes. Its implementation was then extended to two space dimensions [3]. In this paper, we present the formulation of AGEB for the solution of the heat equation in three space dimensions and then describe its parallel implementation on the PVM on a model problem.

2 Formulation of the AGEB Method

Consider the following heat equation,

$$\frac{\partial U}{\partial t} = \frac{\partial^2 U}{\partial x^2} + \frac{\partial^2 U}{\partial y^2} + \frac{\partial^2 U}{\partial z^2} + h(x, y, z, t), \quad (x, y, z, t) \in R \times (0, T] \ , \qquad (1)$$

V.N. Alexandrov et al. (Eds.): ICCS 2001, LNCS 2073, pp. 918–927, 2001.
© Springer-Verlag Berlin Heidelberg 2001

subject to the initial condition

$$U(x,y,z,0) = F(x,y,z), \quad (x,y,z,t) \in R \times \{0\} \ , \tag{2}$$

and the boundary conditions

$$U(x,y,z,t) = G(x,y,z,t), \quad (x,y,z,t) \in \partial R \times (0,T] \ , \tag{3}$$

where R is the cube $0 < x,y,z < 1$ and ∂R its boundary. A generalised approximation to (1) at the point $(x_i, y_j, z_k, t_{N+1/2})$ is given by (with $0 \le \theta \le 1$),

$$\begin{aligned}\frac{u_{i,j,k}^{[N+1]} - u_{i,j,k}^{[N]}}{\Delta t} = \frac{1}{(\Delta x)^2}\left\{\theta(\delta_x^2+\delta_y^2+\delta_z^2)u_{i,j,k}^{[N+1]} + (1-\theta)(\delta_x^2+\delta_y^2+\delta_z^2)u_{i,j,k}^{[N]}\right\} \\ + h_{i,j,k}^{[N+1/2]}, \quad i,j,k = 1,2,\ldots,m \ ,\end{aligned} \tag{4}$$

leading to the seven-point formula,

$$\begin{aligned}&-\lambda\theta u_{i-1,j,k}^{[N+1]} + (1+6\lambda\theta)u_{i,j,k}^{[N+1]} - \lambda\theta u_{i+1,j,k}^{[N+1]} - \lambda\theta u_{i,j-1,k}^{[N+1]} - \lambda\theta u_{i,j,k-1}^{[N+1]} \\ &\quad - \lambda\theta u_{i,j+1,k}^{[N+1]} - \lambda\theta u_{i,j,k+1}^{[N+1]} = \lambda(1-\theta)u_{i-1,j,k}^{[N]} + (1-6\lambda\theta)(1-\theta)u_{i,j,k}^{[N]} \\ &\quad + \lambda(1-\theta)u_{i+1,j,k}^{[N]} + \lambda(1-\theta)u_{i,j-1,k}^{[N]} + \lambda(1-\theta)u_{i,j,k-1}^{[N]} + \lambda(1-\theta)u_{i,j+1,k}^{[N]} \\ &\quad + \lambda(1-\theta)u_{i,j,k+1}^{[N]} + \Delta t h_{i,j,k}^{[N+1/2]} \ .\end{aligned} \tag{5}$$

By considering our approximations as sweeps parallel to the xy-plane of the cube R, (5) can be written in matrix form as,

$$\mathbf{A}\mathbf{u}_{[xy]}^{[N+1]} = \mathbf{f} \ . \tag{6}$$

By splitting $\mathbf{A}$ into the sum of its constituent symmetric and positive definite matrices $\mathbf{G}_1, \mathbf{G}_2, \mathbf{G}_3, \mathbf{G}_4, \mathbf{G}_5$ and $\mathbf{G}_6$ we have

$$\mathbf{A} = \mathbf{G}_1 + \mathbf{G}_2 + \mathbf{G}_3 + \mathbf{G}_4 + \mathbf{G}_5 + \mathbf{G}_6 \ , \tag{7}$$

with these matrices taking block banded structures as shown in Fig. 1.

Using the well-known fact of the parabolic-elliptic correspondence and employing the fractional splitting of Brian [4], the AGEB scheme takes the form,

$$\begin{aligned}(r\mathbf{I}+\mathbf{G}_1)\mathbf{u}_{[xy]}^{(n+1/7)} &= (r\mathbf{I} - (\mathbf{G}_1+\mathbf{G}_2+\mathbf{G}_3+\mathbf{G}_4+\mathbf{G}_5+\mathbf{G}_6))\mathbf{u}_{[xy]}^{(n)} + \mathbf{f} \\ &= ((r\mathbf{I}+\mathbf{G}_1) - \mathbf{A})\mathbf{u}_{[xy]}^{(n)} + \mathbf{f} \\ (r\mathbf{I}+\mathbf{G}_2)\mathbf{u}_{[xy]}^{(n+2/7)} &= r\mathbf{u}_{[xy]}^{(n+1/7)} + \mathbf{G}_2\mathbf{u}_{[xy]}^{(n)} \\ (r\mathbf{I}+\mathbf{G}_3)\mathbf{u}_{[xy]}^{(n+3/7)} &= r\mathbf{u}_{[xy]}^{(n+2/7)} + \mathbf{G}_3\mathbf{u}_{[xy]}^{(n)} \\ (r\mathbf{I}+\mathbf{G}_4)\mathbf{u}_{[xy]}^{(n+4/7)} &= r\mathbf{u}_{[xy]}^{(n+3/7)} + \mathbf{G}_4\mathbf{u}_{[xy]}^{(n)} \\ (r\mathbf{I}+\mathbf{G}_5)\mathbf{u}_{[xy]}^{(n+5/7)} &= r\mathbf{u}_{[xy]}^{(n+4/7)} + \mathbf{G}_5\mathbf{u}_{[xy]}^{(n)} \\ (r\mathbf{I}+\mathbf{G}_6)\mathbf{u}_{[xy]}^{(n+6/7)} &= r\mathbf{u}_{[xy]}^{(n+5/7)} + \mathbf{G}_6\mathbf{u}_{[xy]}^{(n)} \\ \mathbf{u}^{(n+1)} &= \mathbf{u}^{(n)} + 2(\mathbf{u}^{(n+6/7)} - \mathbf{u}^{(n)}) \ .\end{aligned} \tag{8}$$

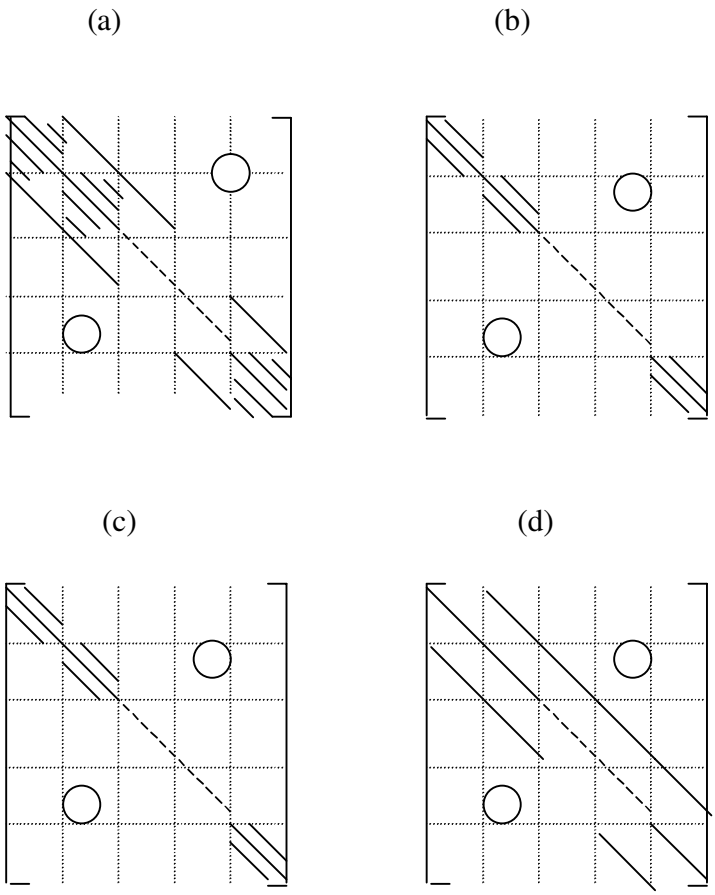


Fig. 1. (a) $\mathbf{A}$, (b) $\mathbf{G}_1+\mathbf{G}_2$, (c) $\mathbf{G}_3+\mathbf{G}_4$, and (d) $\mathbf{G}_5+\mathbf{G}_6$. Note that $\text{diag}(\mathbf{G}_1+\mathbf{G}_2) = \text{diag}(\mathbf{A})/3$, $\text{diag}(\mathbf{G}_3+\mathbf{G}_4) = \text{diag}(\mathbf{A})/3$, $\text{diag}(\mathbf{G}_5+\mathbf{G}_6) = \text{diag}(\mathbf{A})/3$. All are of order $(m^3 \times m^3)$.

The approximations at the first and the second intermediate levels are computed directly by inverting $(r\mathbf{I}+\mathbf{G}_1)$ and $(r\mathbf{I}+\mathbf{G}_2)$. The computational formulae for the third and fourth intermediate levels are derived by taking our approximations as sweeps parallel to the yz-plane. Here, the u values are evaluated at points lying on planes which are parallel to the yz-plane and on each of these planes, the points are reordered row-wise (parallel to the y-axis). Finally, by considering our approximations as sweeps parallel to the xz-plane followed by a reordering of the points column-wise (parallel to the z-axis) enable us to determine the AGEB equations at the fifth and sixth intermediate levels. Note that all solutions at each iterate are generated rather than stored. Hence the actual inverse is not used by the algorithm. The AGEB sweeps involve tridiagonal systems which in turn entails at each stage the solution of (2×2) block systems. The iterative procedure is continued until convergence is reached.

3 Parallel Implementation of the AGEB Algorithm

One must ensure that an effective parallel implementation of the algorithm leads to a substantial increase in the computational count per data exchange, major reduction in synchronisation frequency and subsequent decrease in communication sessions.

A typical parallel implementation involves the assignment of a block of grids to each task to a surface so that each task only communicates with its limited nearest neighbours. Only the top, bottom, left and right surfaces of the block need to be exchanged between neighbouring tasks. As an example, Fig. 2 illustrates the pattern of communication with 4 tasks ($p = 4$) on a linear system of order $m = 32$.

It is also important to maintain load balancing in the distribution of m grids to tasks $P_1, P_2, \ldots, P_p$. The data decomposition of the AGEB algorithm is run asynchronously and simultaneously at every time level, where each task is allocated m/p grids. It proceeds for every task at each time level until the local error and the approximate solution are computed at the last time level. These tasks then send the local errors to the master, which in turn processes the global error.

On the PVM, parallel implementation of AGEB is based on one master and many tasks. The master program is responsible in constructing the m grid sizes, computing the initial values, portioning the grid into blocks of surface, assigning these blocks to the p task modules, distributing the task to different processors and receiving local errors from the tasks. Each block that is assigned to a task module is composed of m/p blocks.

A task process starts computations after it receives a work assignment. A task module $q(q < P)$ performs the AGEB iterations on the grid points of the assigned block which is composed of surfaces with indices between

$$SUR_q(\text{start}) = \frac{m(q-1)}{p} \quad \text{and} \quad SUR_q(\text{end}) = \frac{mq}{p} - 1 \ ,$$

where SUR refers to the surface. The task q will transmit

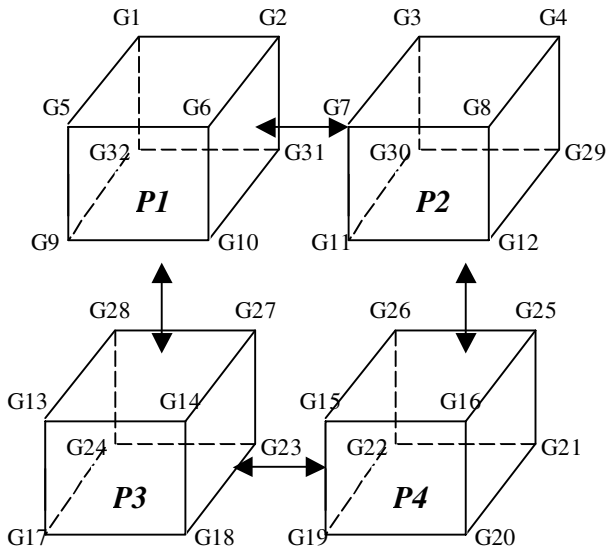


Fig. 2. Communication of data exchange between 4 tasks and 32 grids

i. to its upper neighbour (task $q-1$), $SUR_q(\text{start})$ and receive from it

$$SUR_q(\text{start}) - 1 = SUR_{q-1}(\text{end})$$

ii. to its lower neighbours (task $q+1$), $SUR_q(\text{end})$ and receive from it

$$SUR_q(\text{end}) + 1 = SUR_{q+1}(\text{start}) \ .$$

Since multiple copies of the same task code run simultaneously, the tasks will exchange data with their neighbours at different times. At this point, the barrier function is called by the PVM library routine for synchronisation. The tasks will repeat the above procedure, until the local convergence criterion is met. The definition of the residual computed in the task q is as follows,

$$r[i][j][k] = \max\left\{\left|u_{i,j,k}^{[N+1]} - u_{i,j,k}^{[N]}\right|, \quad (i,j,k) \in q\right\} \ .$$

The tasks will return all its local errors to the master module. After receiving the locally converged blocks from the tasks, the master module checks whether the global convergence is satisfied,

$$r[i][j][k] \leq \epsilon, \quad \forall i,j,k \in [0,m] \ ,$$

where $r[i][j][k] = \max\left\{\left|u_{i,j,k}^{[N+1]} - u_{i,j,k}^{[N]}\right|, \quad (i,j,k) \in q\right\}$.

This procedure is repeated and the system terminates if a global convergence is reached. Otherwise the master repartitions the blocks and reassigns them to the p tasks.

4 Numerical Results and Discussion

The following problem is solved using the AGEB algorithm,

$$\frac{\partial U}{\partial t} = \frac{\partial^2 U}{\partial x^2} + \frac{\partial^2 U}{\partial y^2} + \frac{\partial^2 U}{\partial z^2} + h(x, y, z, t), \quad 0 \leq x, y, z, t \leq 1, \quad t \geq 0 \ ,$$

with

$$h(x, y, z, t) = (3\pi^2 - 1)e^{-t} \sin \pi x \sin \pi y \sin \pi z \ ,$$

subject to the initial condition,

$$U(x, y, z, 0) = \sin \pi x \sin \pi y \sin \pi z \ ,$$

and boundary conditions

$$U(0, y, z, t) = U(1, y, z, t) = U(x, 0, z, t) = U(x, y, 0, t) = U(x, y, 1, t) = 0 \ ,$$

Our PVM platform consists of a cluster of 9 SUN Sparc Classic II workstations each running at a speed of 70 MHz and configured with 32 Mbytes of system memory. The workstations are connected by a 100/10 base T/3com Hub, Super stack II via a Baseline Dual Speed Hub 12 port network. The PVM software only resides in the master processor.

As measures of performance of our algorithm, the following definitions are used:

$$\text{Speed-up ratio} \qquad S_p = T_1/T_p \tag{9}$$

$$\text{Efficiency} \qquad E_p = S_p/p \tag{10}$$

$$\text{Effectiveness} \qquad F_p = S_p/C_p \tag{11}$$

where $C_p = pT_p$, T_1 the execution time on a serial machine and T_p the computing time on a parallel machine with p processors.

Figure 3 shows the speedup factor plotted against the number of processors p. It indicates that high speedups are obtained only for large values of m. An impressive gain in the speedups can be expected for even larger problems. Possible reasons for this are: the relatively high communication time for passing data between the master and slaves compared with the computation time of the AGEB method; wasteful idle time at a barrier synchronisation point before proceeding with the next iteration; the contribution from parallel processing which is less than the number of processors and the contribution from the distributed memory hierarchy, which reduces the time consuming access to virtual memory for large linear equations. However, we are constrained by the relatively small

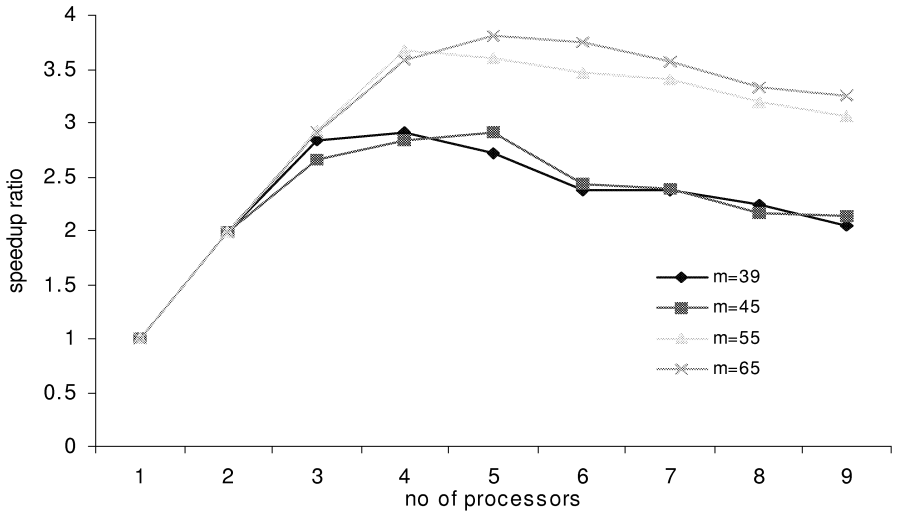


Fig. 3. Speedup ratio vs Number of processors

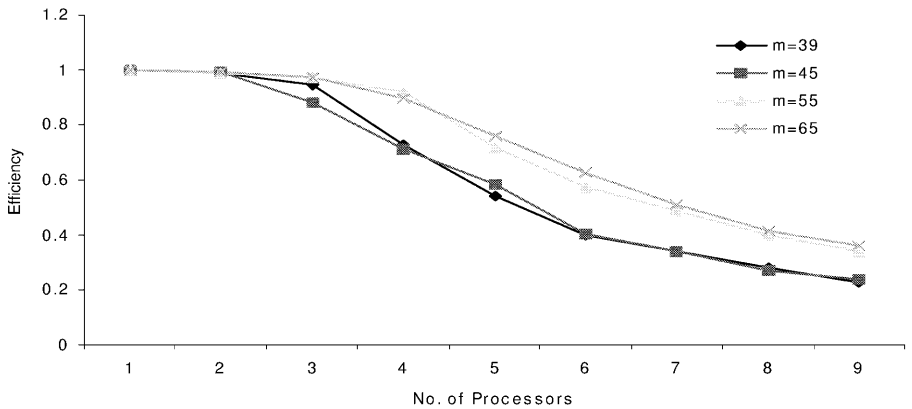


Fig. 4. Efficiency vs Number of processors

memory of the system. The speedup starts to degrade when more than 5 processors are used. It must also be noted that the timing for message passing is relatively slow for inter-processor communication using Ethernet Card 10-based network.

As expected, Fig. 4 depicts that efficiency decreases with increasing p. It deteriorates when more than 3 processors are used. This deterioration is a result of poor load balancing attained when only the small block is spread across more than 3 processors. Hence the high overhead cost due to synchronisation. From (9)–(11),

$$F_p = S_p/(pT_p) = E_p/T_p = E_pS_p/T_1$$

which clearly shows that F_p is a measure both of speedup and efficiency. Therefore, a parallel algorithm is said to be effective if it maximises F_p and hence $F_pT_1 (= S_pE_p)$. From Fig. 5, we see that F_pT_1 has a maximum when $p = 3$ for $m = 39$ which indicates that $p = 3$ is the optimal choice of number of processors. Similarly, we can infer from $m = 45$ and 55 that the optimal choice of number of processors is given respectively by $p = 3$ and $p = 4$ allowing for inconsistencies due to load balancing.

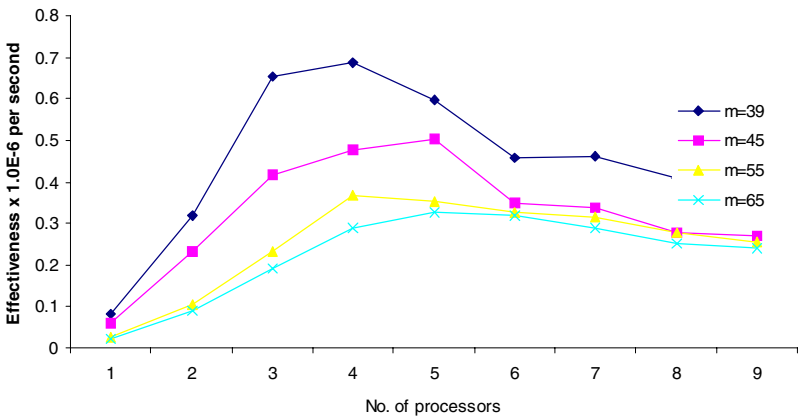


Fig. 5. Effectiveness vs Number of processors

As a general conclusion, the inherently explicit, stable and highly accurate algorithm is found to be well suited for parallel implementation on the PVM where data decomposition is run asynchronously and concurrently at every time level.

5 Conclusions

The stable and highly accurate AGEB algorithm is found to be well suited for parallel implementation on the PVM where data decomposition is run asynchronously and concurrently at every time level. The AGEB sweeps involve tridiagonal systems which require the solution of (2×2) block systems. Existing parallel strategies could not be fully exploited to solve such systems. The AGEB algorithm, however, is inherently explicit and the domain decomposition strategy is efficiently utilised.

The PVM is favoured to MPI as the computing platform because of the flexibility of the former to communicate across architectural boundaries. This is especially relevant since this research project was initially undertaken with the view of utilising readily available heterogeneous cluster of computing resources in our laboratories. Furthermore, we have already noted that higher speedups could be expected for larger problems. Coupled with this is the advantage that the PVM has on its fault tolerant features. Hence, for large real field problems which we hope to solve in the immediate future these features become more important as the cluster gets larger.

Acknowledgments

The authors wish to express their gratitude and indebtedness to the Universiti Kebangsaan Malaysia, Universiti Tenaga Nasional and the Malaysian government for providing the moral and financial support under the IRPA grant for the successful completion of this project.

References

1. Evans, D.J., Sahimi, M.S.: The Alternating Group Explicit Iterative Method (AGE) to Solve Parabolic and Hyperbolic Partial Differential Equations. In: Tien, C.L., Chawla, T.C. (eds.): Annual Review of Numerical Fluid Mechanics and Heat Transfer, Vol. 2. Hemisphere Publication Corporation, New York Washington Philadelphia London (1989)
2. Yanenko, N.N.: The Method of Fractional Steps. Springer-Verlag, Berlin Heidelberg New York (1971)
3. Evans, D.J., Sahimi, M.S.: The Alternating Group Explicit(AGE) Iterative Method for Solving Parabolic Equations I: 2-Dimensional Problems. Intern. J. Computer Math. 24 (1988) 311-341
4. Peaceman, D.W.: Fundamentals of Numerical Reservoir Simulation. Elsevier Scientific Publishing Company, Amsterdam Oxford New York (1977)
5. Geist, A., Beguelin, A., Dongarra, J., Jiang, W., Manchek, R., Sunderam, V., PVM: Parallel Virtual Machine & User's Guide and Tutorial for Networked Parallel Computing. MIT Press, Cambridge, Mass (1994)
6. Hwang, K. and Xu, Z., Scalable Parallel Computing: Technology, Architecture, Programming. McGraw Hill, (1998)
7. Lewis,T.G. and EL-Rewini,H., Distributed and Parallel Computing, Manning Publication, USA, (1998)

8. Quinn, .M.J., Parallel Computing Theory and Practice, McGraw Hill, (1994)
9. Wilkinson,.B. and Allen, M., Parallel Programming Techniques and Applications Using Networked Workstations and Parallel Computers, Prentice Hall,Upper Saddle River, New Jersey 07458 (1999)

A Pollution Adaptive Mesh Generation Algorithm in r-h Version of the Finite Element Method

Soo Bum Pyun and Hyeong Seon Yoo

Department of Computer Science, Inha University,
Inchon, 402-751, South Korea
hsyoo@inha.ac.kr

Abstract: In this paper, we propose a simplified pollution adaptive mesh generation algorithm, which concentrate on the boundary node based on the element pollution error indicator. The automatic mesh generation method is followed by either a node-relocation or a node-insertion method. The boundary node relocation phase is introduced to reduce pollution error estimates without increasing the boundary nodes. The node insertion phase greatly improves the error and the factor with the cost of increasing the node numbers. It is shown that the suggested r-h version algorithm converges more quickly than the conventional one.

1 Introduction

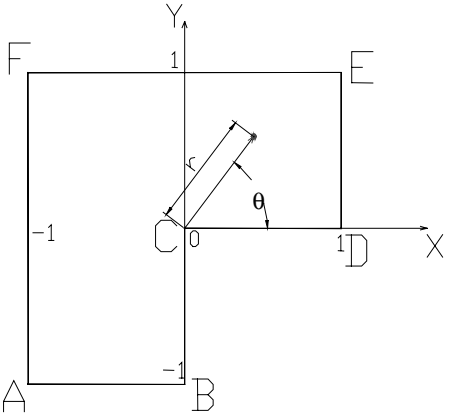
Most engineering problems are described in polygonal domains with geometric singularities. These singularities make the solution diverge to infinity and cause the conventional error estimators to severely underestimate the error in any patch outside the neighborhood of the singular point. Since Babuska's works about error estimators and pollution errors it is known that the pollution error estimates are much more than the local error ones [1,2,3,4]. It was demonstrated that the conventional Zienkiewicz-Zhu error estimator [5,6,7,8] was insufficient and should include a pollution error indicator [1,4]. The pollution-adaptive feedback algorithm employs both local error indicators and pollution error indicators to refine the mesh outside a larger patch, which includes a patch and one to two surrounding mesh layers [2,3]. The conventional pollution adaptive algorithm bisects the element for every iteration and needs a lot of iterations to converge.

We concentrate only on a problem boundary since the singularities exist on the boundary and mesh sizes change gradually regardless of the mesh generation algorithm. A mesh generation algorithm, which uses a node relocation method (r-method) as well as h-method of the finite element method for boundary elements, is proposed. The algorithm employs a boundary-node relocation at first and then does a node insertion based on the pollution error indicator.

V.N. Alexandrov et al. (Eds.): ICCS 2001, LNCS 2073, pp. 928-936, 2001.
© Springer-Verlag Berlin Heidelberg 2001

2 The Model Problem

Consider a typical *L*-shaped polygon, $\Omega \subseteq \mathbf{R}^2$, with boundaries $\partial\Omega = \Gamma = \Gamma_D \cup \Gamma_N$, $\Gamma_D \cap \Gamma_N = \{\}$ where Γ_D is the *Dirichlet* and Γ_N is the *Newmann* boundary (Fig.1).



$\Gamma_D = CD$

$\Gamma_N = AB \cap BC \cap DE \cap EF \cap FA$

Fig. 1. The L-shaped domain for the model problem

We will consider Laplacian with mixed boundary conditions. Let us $H^1_{\Gamma_D} \equiv \{u \in H^1(\Omega) \,|\, u = 0 \quad on \quad \Gamma_D\}$. Then the variational formulation of this model problem satisfies (1).

Find $u_h \in S^p_{h,\Gamma_D}(\Omega) := H^1_{\Gamma_D} \cap S^p_h$ such that

$$B_\Omega(u_h, v_h) = \int_{\Gamma_N} g v_h \quad \forall v_h \in S^p_{h,\Gamma_D} \qquad (1)$$

A patch error was expressed only by a local error, but it was demonstrated that the pollution error should include the patch error . The local error was improved by considering a mesh patch ω_h with a few surrounding mesh layers. The equilibrated residual functional is the same for the local error and the pollution error. But the pollution error was calculated by considering the outside of the larger patch, ω_h.

$$e_h\Big|_{\omega_h} = V_1^{\tilde{\omega}_h} + V_2^{\tilde{\omega}_h} \qquad (2)$$

where, $V_1^{\tilde{\omega}_h}$; local error on $\tilde{\omega}_h$

$V_2^{\tilde{\omega}_h}$; pollution error on $\tilde{\omega}_h$

$\tilde{\omega}_h$; ω_h + a few mesh layers

Let us denote $\|v\|_S = \sqrt{B_S(v,v)}$ energy norm over any domain $S \subseteq \Omega$, then the

equation (3) can be a pollution estimator with $\bar{x} \in \omega_h$, [1,2,3].

$$\left\| V_2^{\tilde{\omega}_h} \right\|_{\omega_h} \cong \sqrt{|\omega_h|} \sqrt{\left(\frac{\partial V_2^{\tilde{\omega}_h}}{\partial x_1}(\bar{x}) \right)^2 + \left(\frac{\partial V_2^{\tilde{\omega}_h}}{\partial x_2}(\bar{x}) \right)^2} \qquad (3)$$

3 The Proposed Algorithm

3.1 The Basic Idea

For adaptive control of the pollution error in a patch of interest, the conventional algorithm fixes meshes in the patch and refines meshes outside the patch especially near singularities. The algorithm calculates an element pollution indicator and regularly divides $\gamma\%$ of elements whose pollution indicators are high [2]. This algorithm is as following Fig.2.

```
Let  T_h = T_h^0
 Compute the finite element solution on T_h , ε_ωh and
M_ωh ;
 While ( M_ωh > t%ε_ωh ) do
      For(each element )do
          Compute μ_τ ,  τ∈ T_h , τ∉ ω̃_h ;
          If ( μ_τ ≥ γ max μ_τ )
              Subdivided τ regularly;
          Endif
      Endfor
    Compute the finite element solution on T_h and
     ε_ωh , M_ωh ;
 Endwhile
```

Fig. 2. Structure of the conventional algorithm

In Fig.2 we denote the element pollution error M_{ω_h}, the local error ε_{ω_h} and the element pollution indicator μ_τ [2]. Since the conventional algorithm bisects the element length, it could be accelerated if we have smaller boundary elements near the singular points. Therefore it is natural to think about combining r and h method.

In our proposed algorithm, we concentrated only on boundary nodes and whole interior area is triangulated automatically by the constrained Delaunay algorithm [9]. Our algorithm employs two ideas for the control of the boundary-nodes. The first is to relocate a boundary node. It makes boundary nodes near a singular point close to the point. The other is to insert a node between the boundary nodes of elements whose pollution indicators are larger than the specified value. In the relocation phase, the

new boundary element length is calculated by using the following relationship between the pollution error estimator and the element size [1,11].

$$\text{Let} \quad \left\| V_2^{\tilde{\omega}_h} \right\|_{\omega_h} \approx h^{2\lambda+1} \tag{4}$$

where λ ; the exponent for singular point.

From this expression, we can deduce old and new element length as following,

$$\left\| V_2^{\tilde{\omega}_h} \right\|_{\omega_h,old} = Ch_{old}^{2\lambda+1} \tag{4'}$$

$$\left\| V_2^{\tilde{\omega}_h} \right\|_{\omega_h,new} = Ch_{new}^{2\lambda+1} \tag{4"}$$

Combining two equations, we obtain h_{new},

$$h_{new} = h_{old} \times \left(\left\| V_2^{\tilde{\omega}_h} \right\|_{\omega_h,old} / \left\| V_2^{\tilde{\omega}_h} \right\|_{\omega_h,new} \right)^{-\frac{1}{2\lambda+1}} \tag{5}$$

In order to get the pollution error smaller than the local error we use $t\varepsilon_{\omega_h} \approx t \left\| V_1^{\tilde{\omega}_h} \right\|_{\omega_h}$ instead of $\left\| V_2^{\tilde{\omega}_h} \right\|_{\omega_h,new}$. t is a user-specified constant between 0 and 1. And $\left\| V_2^{\tilde{\omega}_h} \right\|_{\omega_h,old}$ will be $\mu_\tau \approx \left\| V_2^{\tilde{\omega}_h} \right\|_{\omega_h} / \sqrt{|\omega_h|}$ since the pollution error consists of the element pollution error indicators outside $\tilde{\omega}_h$. Finally the new element size becomes,

$$h_{new} = h_{old} \times (\zeta_\tau)^{-\frac{1}{2\lambda+1}} \tag{6}$$

where $\zeta_\tau \equiv \dfrac{\mu_\tau}{t\varepsilon_{\omega_h}}$

This new element size has an effect on the location of the boundary node, especially the nodes on BC and CD in Fig 1. If the ratio of the element length $(\zeta_\tau)^{-\frac{1}{2\lambda+1}}$ is less than 1, the algorithm moves the node to the singular point. But if it is greater than 1, the new length is discarded and the location of the node remains fixed to have stable solution. This relocation method is for reducing the number of iteration to get the final mesh. The boundary node insertion phase takes part in a high quality of the error estimator, this phase is the same as others [1,2,3].

3.2 The Proposed Algorithm

A binary number Flag is employed to alternate the boundary relocation and the node insertion process. If the flag is 0, the relocation phase is performed. Figure 3 shows the entire procedure.

```
Let  T_h = T_h^0  and set Flag = 0
 Compute the finite element solution on  T_h , ε_{ω_h} and  M_{ω_h} ;
 While ( M_{ω_h} > t%ε_{ω_h} ) do
   Switch ( Flag )
     Case  0 :  /* relocation phase*/
        For ( each element on boundary) do
           Calculate μ_τ ;
           Calculate  ζ_τ  and  h^(k+1)  ;
           If ( (ζ_τ)^(-1/(2λ+1)) < 1.0 )  do
           Relocate the node of the element on boundary;
           Endif
        Endfor
        Set Flag = 1;
        Break ;
     Case  1:   /* node-insertion phase */
        /* The same as Fig.2 */
        Set Flag = 0;
        Break ;
    Endswitch
    Generate mesh using nodes on boundary ;
    Compute the finite element solution on  T_h  and
     ε_{ω_h} , M_{ω_h} ;
 Endwhile
```

Fig. 3. Structure of the proposed algorithm

The algorithm starts with the initial mesh and set Flag 0. The boundary node relocation is controlled by $(\zeta_\tau)^{-\frac{1}{2\lambda+1}}$. If the value is below 1, the element shrinks to singular point. In the node insertion phase, a new node is added on the middle of the boundary element. This r-h method makes fewer nodes on the boundary than the h-version.

The interior mesh generation phase is following the control of nodes on boundaries. This step is performed by the constrained Delaunay method. And the finite element analysis and error estimations are following.

4 Numerical Results and Discussions

We considered the mixed boundary-valued problem for the Laplacian over a L-shaped domain and applied boundary conditions consistent with the exact solution

$u(r,\theta) = r^{\frac{1}{3}} \sin(\frac{1}{3}\theta)$ [1]. An interior patch element ω_h far from the singular point as in Fig.4 is chosen. The L-shaped domain is meshed by uniform quadratic triangles (p=2) with h=0.125. In table.1, we show the numerical results for the model problem. Though the local error estimate (ε_{ω_h}) is almost constant, the pollution error decreases dramatically with iteration in the r-h version.

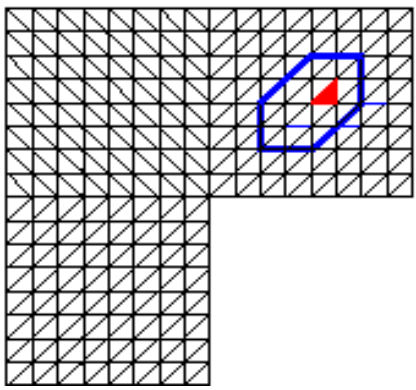


Fig. 4. The initial mesh for numerical example
ω_h : patch, a shaded triangular element
ω_h : large patch, elements enclosed by thick hexagonal line

Table 1. Results of the model problem

Iter.	ε_{ω_h} x E-05		M_{ω_h} x E-05	
	Fig.2	Fig.3	Fig.2	Fig.3
0	7.43	7.43	66.72	66.72
1	7.75	8.29	44.19	21.90
2	7.96	8.33	29.68	7.00
3	8.09	8.33	20.62	6.09
4	8.20	8.32	13.96	2.88
5	8.11		10.01	
6	8.12		7.28	
7	7.99		5.43	
8	8.00		4.45	
9	8.01		3.88	
10	8.01		3.55	
11	8.02		3.34	
12	8.02		3.26	
13	8.02		3.14	
14	8.02		3.13	
15	8.02		3.06	

After the second iterations in h version, the pollution error reduces about half of the initial value. But the pollution error of r-h version is decreased to one of ninth. This significant reduction makes the number of iteration less than that of the conventional one. The pollution factor is defined by the ratio of a pollution error estimate and the local error estimate, $\rho_\tau = M_{\omega_h} / \varepsilon_{\omega_h}$. In Fig.5 we can see that the pollution factor decrease more rapidly for the proposed algorithm case. In case of the proposed algorithm the pollution factor becomes less than 0.4 only after 4 iterations. From this result, we note that the proposed algorithm controls the pollution error and is effective. The total number of iteration is 4, which is much smaller than one of the conventional algorithms. In Fig. 6 we show the final mesh, which is obtained by the proposed algorithm.

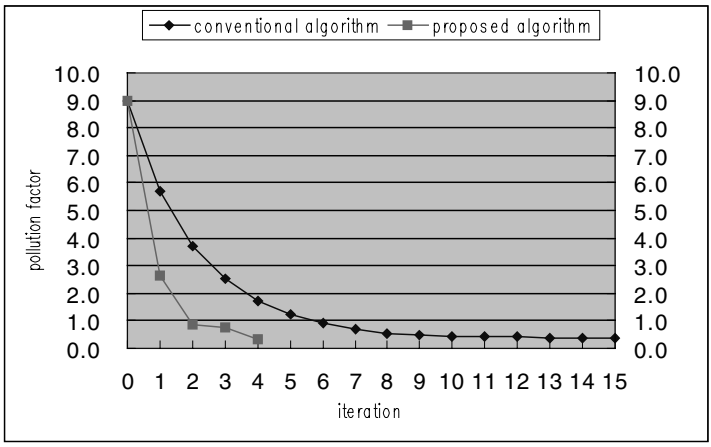


Fig. 5. The pollution factor, β_τ versus iteration

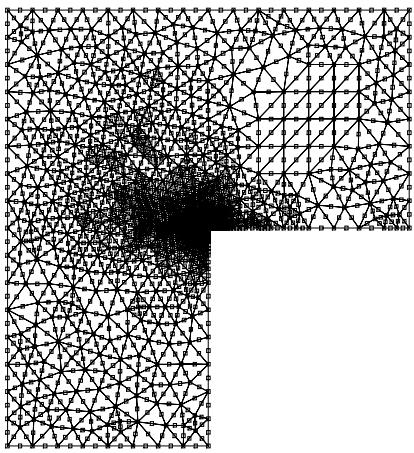


Fig. 6. The final mesh after 4 iterations by the proposed algorithm (N=3493,E=1696)

Table.2 shows the numerical results of two algorithms. From the table we knows that the proposed algorithm needs about 21.03% less time than the conventional one. It causes from more iteration of the conventional algorithm. Therefore the proposed algorithm has more effective results than the conventional one.

Table 2. Results of the model problem

	Conventional algorithm	Proposed algorithm
No. of iteration	15	4
No. of node	2597	3493
No. of element	1242	1696
Computation time	5462 sec.	4308 sec.

5 Conclusions

The pollution factor shows that the proposed algorithm converges after only 4 iterations comparing with 15 of the conventional one. The proposed r-h algorithm is easy to handle since it considers only the boundary elements. The boundary node-relocation phase is very effective for this fast convergence.
The pollution error estimates is improved from 66.72 to 2.88. Though the number of node is increased from 2597 to 3493, the total calculation time has been improved due to the decrease of the number of iteration. It is proved that the well known Delaunay method in this pollution adaptive algorithm is effective.

References

1. I. Babŭska , T. Strouboulis, A. Mathur and C.S. Upadhyay, “Pollution error in the h-version of the finite element method and the local quality of a-posteriori error estimates”, Finite Elements Anal. Des.,17,273-321(1994)
2. I. Babŭska , T. Strouboulis, C.S. Upadhyay and S.K. Gangaraj, “A posteriori estimation and adaptive control of the pollution error in the h-version of the finite element method”, Int. J. Numer. Method Engrg., 38, 4207-4235(1995)
3. I. Babŭska , T. Strouboulis, S.K. Gangaraj, “Practical aspects of a-posteriori estimation and adaptive control of the pollution error for reliable finite element analysis”, http://yoyodyne.tamu.edu/research/pollution/index.html(1996)
4. I. Babŭska , T. Strouboulis, S.K. Gangaraj and C.S. Upadhyay, “Pollution error in the h-version of the finite element method and the local quality of the recovered derivatives”, Comput. Methods Appl. Mech. Engrg.,140,1-37(1997)
5. O.C. Zienkiewicz, and J.Z.Zhu, “The Superconvergent Patch Recovery and a posteriori estimators. Part1. The recovery techniques”, Int. Numer. Methods Engrg.,33,1331-1364(1992)

6. O.C. Zienkiewicz, and J.Z.Zhu, “The Superconvergent Patch Recovery and a posteriori estimators. Part2. Error estimates and adaptivity”, Int. J. Numer. Methods Engrg.,33,1365-1382(1992)
7. O.C. Zienkiewicz, and J.Z.Zhu, “The Superconvergent Patch Recovery(SPR) and adaptive finite element refinement”, Comput. Methods Appl. Mech. Engrg.,101,207-224(1992)
8. O.C. Zienkiewicz, J.Z.Zhu and J. Wu, “Superconvergent Patch Recovery techniques – Some further tests”, Comm. Numer. Methods Engrg., Vol. 9,251-258(1993)
9. B. Kaan Karamete, User manual of 2D Constrained Mesh Generation Mesh2d. http://scorec.rpi.edu/~kaan/mesh2d.tar
10. B.K. Karamete, T. Tokdemir and M. Ger, “Unstructured grid generation and a simple triangulation algorithm for arbitrary 2-D geometries using object oriented programming”, Int. J. Numer. Methods Engrg., 40,251-268(1997)
11. I. Babŭska , T. Strouboulis, and S.K. Gangaraj, “A posteriori estimation of the error in the recovered derivatives of the finite element solution”, Comput. Methods Appl. Mech. Engrg.,150,369-396(1997)

An Information Model for the Representation of Multiple Biological Classifications

Neville Yoon and John Rose

University of South Carolina, Department of Computer Science and Engineering, Columbia, South Carolina 29208 USA

Abstract. We present a model for representing competing classifications in biological databases. A key feature of our model is its ability to support future classifications in addition to current and previous classifications without reorganizing the database. Data in biological databases is typically organized around a taxonomic framework. Biological data must be interpreted in the context of the taxonomy under which it was collected and published. Since taxonomic opinion changes frequently, it is necessary to support multiple taxonomic classifications. This is a requirement for providing comprehensive responses to queries in databases that contain data reflecting incompatible taxonomic classifications.

1 Introduction

Biological taxonomy provides the organizational framework by which biological information is stored, retrieved, and exchanged. Electronic databases represent a relatively new medium for the storage of biological information, but the concepts and labels that people use to interact with them are the same taxa and names used in the taxonomic literature. It is therefore necessary for biological databases to accurately represent these taxonomic constructs.

Unfortunately there is no single, correct classification that categorizes all organisms for all time. Because of the continuous nature of evolution, taxon delimitation is largely arbitrary; there can be no "correct" classifications, only more or less useful ones. Opinions as to what is more useful vary and frequently change as new specimens are collected, new characters are examined, and new analytical techniques are adopted. Consequently the classifications by which biological information is recorded are often replaced. Furthermore, at any one point in time there may be several incompatible classifications competing for acceptance. Biological databases should be capable of reflecting these competing taxonomic hypotheses. Databases unable to do so risk obsolescence.

As a result of changing classifications, it is often the case that many different names have been applied at different times to a particular group of organisms. Conversely, a single name may have been applied to different sets of organisms. Consequently much biological information is associated in the literature with names that are considered incorrect under current classifications. In order to interpret this information in a modern context, one must know not only the classification assumed by the original publication, but also the nomenclatural and taxonomic changes that relate that classification to the current one.

V.N. Alexandrov et al. (Eds.): ICCS 2001, LNCS 2073, pp. 937–946, 2001.
© Springer-Verlag Berlin Heidelberg 2001

Most biological database designs ignore this complexity by using only taxon names to identify taxa [1,2,3,8,10,11,12]. With this approach incompatible classifications cannot be maintained simultaneously because there is no way to distinguish among different taxon concepts with the same name [5,6]. With careful editing by taxonomic experts, such a database can be made to conform to a particular set of complimentary classifications. However, the information stored within cannot easily be made to conform to any conflicting classification, making the database inflexible with respect to changes in taxonomic opinion. Worse, if information is widely compiled from the literature and stored uncritically by name, incompatible taxon concepts will become confounded resulting in a database that does not conform to any classification. A more flexible approach to managing taxonomic information is required.

A system for managing multiple classifications will have to satisfy at least two criteria: First, information must be indexed by specific taxonomic interpretations of names (named taxon concepts) rather than by names alone. Second, there must be a way to represent relationships of shared content among named taxon concepts. This allows information to be aggregated both within classifications through the taxonomic hierarchy and among classifications according to overlap in taxon boundaries.

In the following sections we briefly review four previously proposed designs for the management of multiple classifications in biological databases. Each of these meets the above criteria at least in part. However, none is completely satisfactory for use in databases of descriptive, non-specimen-based biological information intended for non-taxonomists. We derive from them an information model that we believe to be more appropriate for these types of systems.

2 Prior Approaches to Modeling Multiple Classifications

To our knowledge, four information models designed to accommodate incompatible classifications have been published or otherwise made publicly available. These are the Association of Systematics Collections (ASC) draft datamodel [4], the HICLAS model [5,9,15], the International Organization of Plant Information (IOPI) model [6,7], and the Prometheus model [12]. We will discuss each of these using an extended entity-relationship (ER) data modeling vocabulary and notation (Fig. 1).

In the ASC model, both taxon names and taxon concepts are explicitly represented as entity types (Fig. 1). The TAXON CONCEPT (TC) entity type represents a taxonomic circumscription with no inherent name. The many-to-many relationship between names and taxon concepts is resolved through the use of an associative entity type, TAXON-NAME-USE. This traditional ER approach satisfies our first criterion of representation of multiple taxon concepts for the same name. The taxonomic placement of a taxon underneath a superior taxon in a classification is represented by an entity of the recursively associative entity type, TAXONOMIC RELATIONSHIP . This structure implements a network of TCs in which each TC is the root of a taxonomic sub-tree, the leaves of which are species-level

TCs. The circumscription of a TC is the full set of terminal TCs in its sub-tree. This model permits the comparison of TCs according to shared content and thus allows aggregation of information both within and among classifications. However, separating out the complete set of TCs and associated TAXON NAME USEs for a particular classification requires additional constructs not presented in the ASC draft datamodel.

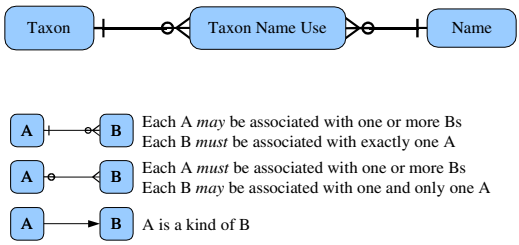


Fig. 1. A partial entity-relationship diagram of the many-to-many relationship between taxa and names

The HICLAS and IOPI models represent circumscriptions implicitly by association with a publication rather than explicitly with an entity type. This approach models taxon concepts as a relationship between a taxon name and a publication and is analogous to the TAXON-NAME-USE entity type of the ASC model without the associated TC. This is called a *taxon view* in the HICLAS model and a *potential taxon* in the IOPI model. We will use the term *name-use* to represent the generalized concept. A name can be used in different contexts in different publications, but by linking information to the name-use, these different taxon concepts are not confused. However, since circumscriptions are not directly represented, the synonymy of different names applicable to the same taxon must be recorded directly rather than by association with overlapping circumscription. The two models take different approaches to this problem.

In the HICLAS model both classificatory relationships and derivational relationships among taxon views are represented. Classifications are represented by trees consisting of taxon views connected by classification relationships. The derivational history of a taxon concept is represented by a set of operation trees that trace the previous taxon concepts that have been split, merged, moved, or accepted to create the taxon view in question. Relationships of shared content can be inferred from these operation trees.

The HICLAS model provides a simple mechanism for the simultaneous maintenance of multiple classifications, for the structural comparison of different classifications, and for tracing the histories of taxon concepts. However, scientific names themselves are treated in a simplified manner that does not allow a complete representation of purely nomenclatural relationships.

The IOPI approach uses an associative entity type called STATUS ASSIGN-

MENT to represent the relationships among potential taxa. Different types of status assignments are used to indicate nomenclatural relationships and relationships of shared content.

The position of a potential taxon in a classification is represented through a simple recursive relationship on the POTENTIAL TAXON (PT) entity type. The restriction that a POTENTIAL TAXON can have only one taxonomic position results in considerable proliferation of PTs.

A sub-tree composed of all of the descendents of a particular PT will often contain PTs representing different circumscriptions for the same taxon name. Berendsohn [6] proposes a simple ranking of taxonomic reference works to resolve these conflicts. When conflicts are found in the generation of a classification tree from the database, those established by the preferred reference works are chosen for presentation. This seems to be a limited and unwieldy method for reconstructing alternative classifications from the database.

The Prometheus model represents biological taxonomy more accurately than any other model published to date. Aspects of biological nomenclature are carefully separated from those related to circumscriptions and classifications to reflect the way that taxa are actually created and named in taxonomic practice.

Taxonomic names are represented by the NOMENCLATURAL TAXON (NT) type. The NT is the combination of a taxon name, a rank, a superior NT for names at species-level ranks, a publication, and a nomenclatural type, which may be a specimen or another NT. Official declarations of nomenclatural status that may affect the priority of a name may also be assigned to NTs. Taxon concepts are represented by CIRCUMSCRIBED TAXON objects which are the combination of either an NT or an informal name, a circumscription, a rank, an author, and possibly a publication.

The Prometheus model is unique in that relationships of shared content are not represented declaratively, but rather are derived from rigorous and detailed representations of taxon content. Taxon content is represented by the CIRCUMSCRIPTION type. A CIRCUMSCRIPTION object specifies a complete set of specimens included in a taxon. For published taxa, the circumscription consists of all the specimens cited in the published description. For experimental taxa, the specimens include all those deemed to belong by the practicing taxonomist. Relationships of synonymy by shared content are derived by directly comparing circumscriptions among CTs. Furthermore, the nomenclatural principles of priority and typification can be applied algorithmically to validate the assignments of taxonomic names to CTs.

This strict, specimen-based approach to comparing taxa is very powerful when complete sets of included specimens are available. In this case all objective relationships of shared content can be found. However, in many cases complete sets of specimens are unavailable or the effort of compiling them exceeds the abilities of a database team. Furthermore, the Prometheus model prohibits the extrapolation of taxonomic inference beyond that directly supported by the specimen content of circumscriptions. The result of these restrictions is that in some information systems, large amounts of information stored within may not

be interpreted with respect to specific classifications and cannot be aggregated according to suspected relationships of shared content. This is not a criticism of the Prometheus approach, which is logically correct. Nonetheless we believe these restrictions may be excessive for many information systems.

3 The PeroBase Model

Our motivation for developing a new information model is to provide the taxonomic framework for PeroBase, an encyclopedic database that manages information on the biology of peromyscine mice for users ranging from the interested layperson to specialists in *Peromyscus biology*. Peromyscine taxonomy has undergone a few major revisions and is still in flux, so we need a model that can represent incompatible classifications. In studying the four information models discussed above, we concluded that the ASC and HICLAS models were not sufficiently complete to handle our nomenclatural information. We are most impressed by the Prometheus model, but do not have the resources to compile the needed specimen lists. Furthermore, we desire the ability to organize information according to a fully-connected taxonomic hierarchy that integrates numerous lower level classifications. This is prohibited in the Prometheus model. We therefore have derived a model that we believe to be more appropriate for our purposes by relaxing the restrictions of the Prometheus model and incorporating ideas from the other models.

3.1 Nomenclature

Scientific taxon names are represented by the NOMENCLATURAL TAXON (NT) entity type of the Prometheus model with minor modifications primarily to accommodate differences between the International Code of Zoological Nomenclature (ICZN) [13] and the International Code of Botanical Nomenclature [14](Fig. 2). As in Prometheus, an NT is the combination of a name element, a taxonomic rank, a taxonomic placement for species ranked NTs, a publication, and a name-bearing type. Our NT differs primarily in the way nomenclatural status is assigned. These assignments are made through the NOMENCLATURAL STATUS ASSIGNMENT entity type analogous to the way nomenclatural status is assigned to potential taxa in the IOPI model. NOMENCLATURAL STATUS ASSIGNMENTs are used to record formal published acts that affect the application of the principle of priority in determining valid names. This usually involves suppression of senior synonyms or homonyms in favor of junior names in prevailing use. In these cases, the preferred NT is also associated with the NOMENCLATURAL STATUS ASSIGNMENT. The publications in which these assignments are made are recorded as associations of NOMENCLATURAL STATUS ASSIGNMENTs with PUBLICATIONS. When status assignments affect the priority of groups of secondary homonyms or heterotypic synonyms, the suppression is only effected while the types of the NTs are considered to fall within the same circumscription. When this is not the

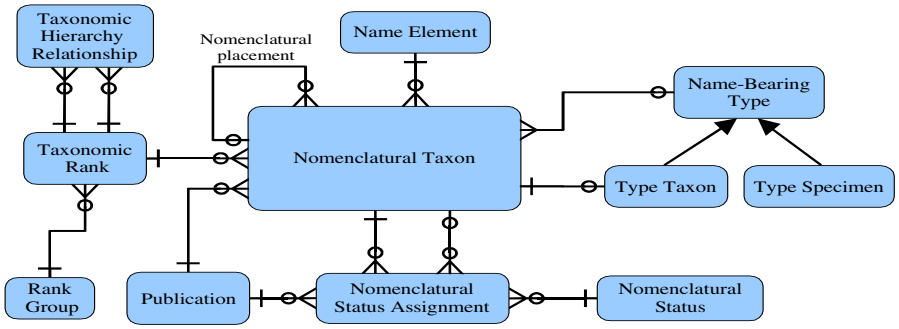


Fig. 2. The NOMENCLATURAL TAXON and associated entity types

case, these status assignments are to be ignored by the system for the purpose of determining valid names.

Every NT has a taxonomic rank represented by an associated TAXONOMIC RANK entity. The ICZN defines three groups of taxonomic ranks over which it claims authority: the species-group, the genus-group, and the family-group. Nomenclatural rules apply differently to nomenclatural taxa depending on the group to which they belong. These taxonomic rank groups are represented in the PeroBase model with TAXONOMIC RANK GROUP entities. Taxonomic ranks are ordered into taxonomic hierarchies to which classifications adhere. The names and order of many ranks are considered obligatory by convention; however, most ranks are optional and taxonomists are free to insert any number of additional ones. This network of superior-subordinate relationships between taxonomic ranks is represented through the TAXONOMIC HIERARCHY RELATIONSHIP entity type. The particular associations represented constrain the taxonomic hierarchies that are permissible in the database.

An NT is associated with a single NAME ELEMENT, but is intended to represent a full scientific name. For taxa above the species rank, the two are the same. However by the principle of binominal nomenclature, the name element of taxa of species and subspecies ranks must be prefixed by the full name of the taxon under which they are placed. A recursive relationship implements the nomenclatural placement of species group NTs permitting the composition of their full names.

Every nomenclatural taxon of family-group rank or below has an actual or potential name-bearing type according to the ICZN. Correspondingly in PeroBase, every NT of family-group rank or below must be associated with a NAME-BEARING TYPE (NB-TYPE). A NB-TYPE may represent either a type specimen for species group taxa, a type species for genus group taxa, or a type genus for family group taxa. To model this, two subtypes of NB-TYPE are used: TYPE SPECIMEN and TYPE TAXON. A TYPE SPECIMEN need not have any attributes in the database, but a TYPE TAXON must have an additional relationship with

an NT that specifies the taxon that is the type species or type genus. Note that NB-TYPE represents only the real or potential existence of a type. While information about real type specimens could be attached to NB-TYPEs, this is not necessary. This "virtual" type concept serves primarily as a reference tool for the determination of synonym and homonym relationships among taxa. This corresponds to the use of "dummy" types in the Prometheus model.

3.2 Classification

The association of a name with a particular circumscription is represented by a TAXONOMIC TAXON (TT) entity (Fig. 3). Both classifications and circumscriptions in the PeroBase model are implemented as trees of TTs whose edges are TAXONOMIC RELATIONSHIP entities. TAXONOMIC RELATIONSHIPs may represent either the classification of a TT under a TT of superior rank (placement), or the inclusion of a monotypic species-level TT in the circumscription of a polytypic species-level TT (inclusion).

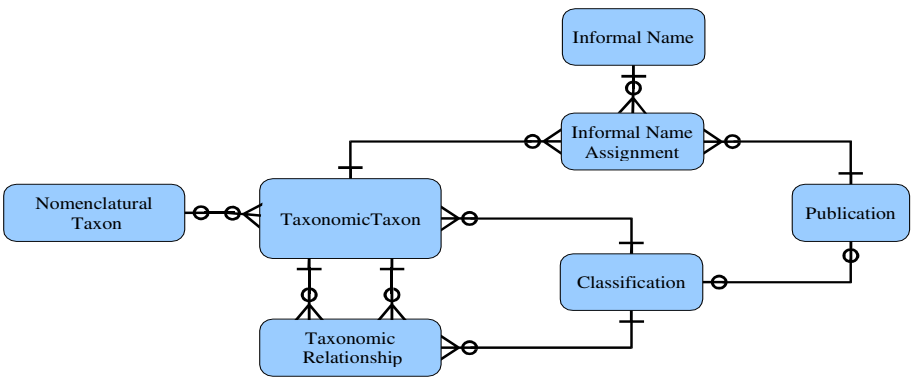


Fig. 3. The TAXONOMIC TAXON and related entity-types

Every TT and every TAXONOMIC RELATIONSHIP is created in the context of a particular classification and must be associated with a CLASSIFICATION entity. The CLASSIFICATION entity is merely a label for use in grouping TTs and TAXONOMIC RELATIONSHIPS to form classification systems. The classification system itself is represented by a tree of TTs whose edges are all placement TAXONOMIC RELATIONSHIPs linked to the CLASSIFICATION.

A circumscription is represented as a set of all NB-TYPES included within the boundaries of a TT. This set is obtained for a TT by finding the species-level TTs that are leaves in any classification tree rooted at that TT and adding for each polytypic leaf TT all monotypic species-level TTs that are related through inclusion type TAXONOMIC RELATIONSHIPs.

A classification may include TTs established in previous classifications as long as neither their associated NTs nor their full set of included types are changed. For example, a classification that rearranges existing species into new subgenera without changing the contents or generic placements of those species can use existing TTs for those species. If an existing taxon is moved from its current position to another, the contents of both its former and new parents have changed, and new TTs are required to represent the new circumscriptions. However, if the former and new parent taxa are themselves both placed under the same TT, no new TT is required for that common parent since its circumscription has not changed. A change in the rank of a taxon requires a new NT and therefore a new TT. A change in the placement of a taxon of species level rank results in a mandatory name change, requiring a new NT and therefore a new TT as well.

The TT representing a newly described species or subspecies may be placed under a previously existing TT even though the circumscription of the parent has expanded to include the new type. Without this exception, the addition of a new type would require new TTs for all superior taxa on all paths to the roots of all classifications that include the new taxon [4]. The different circumscriptions for the same TT are still separable since the original and new classificatory relationships are associated with different CLASSIFICATIONs. The full circumscription for a TT, including all subsequent additions, can be obtained for any point of view by adding the relevant classificatory relationships of subsequent classifications to those of the original definition.

3.3 Synonyms

Homotypic, heterotypic, and most *pro-parte* synonyms for a particular taxonomic taxon can all be found algorithmically as in the Prometheus model. Homotypic synonyms are simply different nomenclatural taxa of the same rank with the same name-bearing type. Heterotypic synonyms for a particular taxonomic taxon are the names of all of the TYPE TAXON OR TYPE SPECIMEN entities included in the taxon's circumscription. The correct name for a particular taxonomic taxon can be found from among its heterotypic synonyms by identifying the one that was published first after eliminating those specified as invalid in associated nomenclatural status assignments.

Most *pro-parte* synonyms for a particular taxonomic taxon can be found by finding all taxonomic taxa in the same rank group that contain any of that taxon's included type specimens. As Pullan, et. al. [12] point out, this will not identify all *pro-parte* synonyms because some taxa may share specimens without sharing any types. We suspect that this level of resolution will rarely be necessary for taxon-based information systems; however, such instances of overlap could be indicated with a new type of taxonomic relationship if so desired.

3.4 Determinations: Assigning Data to Taxa

Descriptive information can be applied to both NTs and TTs. Very often in the biological literature descriptive information is attributed to a taxon identified

by name without specification of the classification assumed. This information is therefore name-based only and can be assigned with full confidence only to NTs [12]. The name under which descriptive information is originally published is recorded in the PeroBase model through an association with an NT via a NOMENCLATURAL TAXON DETERMINATION. We agree with Berendsohn [5] that in many cases information not derived from identified specimens may still be attributable to particular taxon concepts with reasonable confidence. In the PeroBase model, data is associated with a TT through a TAXONOMIC TAXON DETERMINATION entity. The person responsible for the determination and the date on which it was made are both recorded with the determination so that corrected assignments can be made without erasing the history of previous assignments.

4 Conclusion

Accurate information models of biological taxonomy are difficult to design due to the inherent complexity of taxonomic data and nomenclature. As a result, most biological databases have been developed from overly simplistic representations of taxonomy. This is unfortunate because over time the information in these databases will no longer reflect current taxonomic opinion. Keeping these databases up-to-date will require periodic large-scale overhauls, work that could have been largely avoided through the use of a more flexible taxonomic data model. Databases intended to manage descriptive biological information for non-taxonomists tend to have the most simplistic models of taxonomy, yet they may have the greatest need for taxonomic flexibility.

Four models have previously been proposed to permit the simultaneous management of multiple biological classifications, a primary requisite for adaptable biological databases. While sharing many similarities, each of the models has taken a unique approach to solving the problem. These differences reflect slightly different priorities and intended uses and the models have succeeded to various degrees. In our opinion, the most accurate and powerful representation of taxonomy to date is the Prometheus model [12], but large amounts of specimen-level data must be compiled to realize the full strengths of that model. For many kinds of biological information systems this may not be practical. A new model is needed to approach that level of taxonomic flexibility in databases that deal with information above the specimen level of resolution. We have presented a new model of taxonomy derived from the Prometheus model for this purpose. We believe the new model offers the best compromise so far proposed between accuracy and flexibility on one hand and practical applicability on the other.

Our model has been developed to serve as the taxonomic framework for PeroBase, a multi-disciplinary descriptive database of information pertaining to peromyscine mice. Current work involves the implementation of the model with taxonomic information drawn from the literature for this group.

Acknowledgement

This work was funded by NSF grants DBI-9723223 and DBI-9807881.

References

1. Allkin, R., Bisby, F.A. Databases in Systematics. Academic Press, New York (1984)
2. Allkin, R., White, R.J.: Data management models for biological classification. In: Bock, H.H. (ed.): Classification and Related Methods of Data Analysis. Elsevier Science Publishers B.V., North-Holland. (1988) 653–660
3. Allkin, R., White, R.J., Winfield, P.J.: Handling the taxonomic structure of biological data. Mathematical and Computer Modelling 16:6/7 (1992) 1–9
4. Association of Systematics Collections: An Information Model for Biological Collections (Draft) March 1993 version: Report of the Biological Collections Data Standards Workshop August 18–24, 1992. Association of Systematics Collections. Available from: gopher://kaw.keil.ukans.edu/11/standards/asc (1993)
5. Beach, J.H., Pramanik, S., Beaman, J.H.: Hierarchic Taxonomic Databases. In: Fortuner, R. (ed.): Advances in Computer Methods for Systematic Biology. Johns Hopkins University Press, Baltimore (1993) 241–256
6. Berendsohn, W.G.: The concept of "potential taxa" in databases. Taxon **44** (1995) 207–212
7. Berendsohn, W.G.: A taxonomic information model for botanical databases: the IOPI model. Taxon **46** (1997) 283–309
8. Blum, S.D. (ed.): Guidelines and Standards for Fossil Vertebrate Databases: Results of the Society of Vertebrate Paleontology Workshop on Computerization. November 1–4, 1989; Austin, Texas (1991)
9. Jung, S., Perkins, S., Zhong, Y., Pramanik, S., Beaman, J.: A new data model for biological classification. CABIOS **11**:3 (1995) 237–246
10. Krebs, J., Kaesler, R., Chang, Y-M, Miller, D., Brosius, E.: PaleoBank: a Relational Database for Invertebrate Paleontology: Data Model. Paleontological Institute, U. Kansas. http://history.cc.ukans.edu/~paleo. (1996)
11. Pankhurst, R.J.: Taxonomic databases: the PANDORA system. In: Fortuner, R. (ed.): Advances in Computer Methods for Systematic Biology. Johns Hopkins University Press, Baltimore (1993) 230–240
12. Pullan, M.R., Watson, M.F., Kennedy, J.B., Raguenaud, C., Hyam, R.: The Prometheus taxonomic model: a practical approach to representing multiple classifications. Taxon **49** (2000) 55–75
13. International Commission on Zoological Nomenclature: International Code of Zoological Nomenclature (4th ed.). U.California Press, Berkeley (1985)
14. Greuter, W., Barrie, R.R., Burdet, H.M., Chaloner, W.G., Demoulin, V., Hawksworth, D.L., Jorgensen, P.M., Nicholson, D.H., Silva, P.C., Trehane, P., MacNeill, J. (eds.): International Code of Botanical Nomenclature (Tokyo Code). Regnum Veg (1994) 1–389
15. Zhong, Y., Jung, S., Pramanik, S., Beaman, J.H.: Data model and comparison and query methods for interacting classifications in a taxonomic database. Taxon **45** (1996) 223–241

A Precise Integration Algorithm for Matrix Riccati Differential Equations

Wan-Xie Zhong[1] and Jianping Zhu[2]

[1] State Key Laboratory of Structural Analysis for Industrial Equipment, Dalian University of Technology, Dalian 116023, China
[2] Department of Mathematics & Statistics, Mississippi State University
Mississippi State, MS 39762, USA
jzhu@math.msstate.edu

Abstract. An efficient precise integration method for solving the matrix Riccati differential equation is described in this paper. The method is based on repeated combination of extremely small time intervals, which leads to solutions with an accuracy within the machine precision.

1 Introduction

The general matrix Riccati differential equation can be written as

$$\dot{\mathbf{S}} = -\mathbf{B} + \mathbf{SA} - \mathbf{CS} + \mathbf{SDS} \tag{1}$$

where $\mathbf{S}(t)$ is an $m \times n$ matrix to be solved, $\dot{\mathbf{S}}$ is the derivative of $\mathbf{S}(t)$ with respect to t, and $\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{D}$ are all given matrices with dimensions $n \times n$, $m \times n$, $m \times m$, $n \times m$, respectively. The solution of the matrix Riccati differential equation is very important in various applications, such as in optimal control theory, wave propagation, structural mechanics, and game theory [1-4]. The integration domain is $0 \le t \le t_{\mathrm{f}}$, where t_{f} is given, and the boundary condition is given by

$$\mathbf{S}(t_{\mathrm{f}}) = \mathbf{S}_{\mathrm{f}}, \qquad \text{for } t = t_{\mathrm{f}}, \tag{2}$$

where $\mathbf{S}_{\mathrm{f}}$ is given. Note that the integration of (2) goes backward from t_{f} to 0.

The respective dual Riccati differential equation can be written as

$$\dot{\mathbf{T}} = -\mathbf{D} - \mathbf{TC} + \mathbf{AT} + \mathbf{TBT}, \tag{3}$$

where $\mathbf{T}(t)$ is an $n \times m$ matrix to be solved with the initial condition

$$\mathbf{T}(0) = \mathbf{G}_0. \tag{4}$$

Since both equations (1) and (3) are nonlinear, it is very difficult, if not impossible, to find analytical solutions for application problems. The most commonly used solution methods are numerical integration schemes based on finite difference [1,5]. The application of these schemes can be difficult when very high accuracy is desirable, or

V.N. Alexandrov et al. (Eds.): ICCS 2001, LNCS 2073, pp. 947-956, 2001.
© Springer-Verlag Berlin Heidelberg 2001

when the solution changes dramatically (caused by large matrix $\mathbf{S}_{\mathrm{f}}$ at the boundary, for example), or else when the problems being solved are stiff. In this paper, an efficient and accurate scheme for solving Riccati differential equations will be presented. The new scheme is based on the precise time integration method for systems of linear differential equations [6-8]. It can provide accurate numerical solutions to equation (1) with errors in the order of computer round-off errors.

2 Linear Equations and Boundary Conditions

The n -dimensional linear transport process can be described [1,2] by

$$\dot{\mathbf{q}} = \mathbf{A}\mathbf{q} + \mathbf{D}\mathbf{p}, \qquad \dot{\mathbf{p}} = \mathbf{B}\mathbf{q} + \mathbf{C}\mathbf{p} \tag{5}$$

where $\mathbf{q}, \mathbf{p}$ are vectors of dimension *n* and *m*, respectively. When $n = m$, $\mathbf{C} = -\mathbf{A}^{\mathrm{T}}$, and $\mathbf{B}, \mathbf{D}$ are non-negative symmetric matrices, equation (5) becomes the dual equation of continuous time optimal control problem. In general, most problems require $m + n$ boundary conditions corresponding to (5) in the form of

$$\mathbf{q}(0) = \mathbf{q}_0, \quad \text{when} \quad t = 0; \quad \mathbf{p}(t_{\mathrm{f}}) = \mathbf{p}_{\mathrm{f}} \quad \text{when } t = t_{\mathrm{f}}, \tag{6}$$

where $\mathbf{q}_0, \mathbf{p}_{\mathrm{f}}$ are given vectors of dimensions *n* and *m*, respectively. The precise time integration method in [6] was for initial value problems, and those in [7,8] were for conservative systems with $m = n$. The present paper will discuss the precise time integration method for two point boundary value problems in the form of (5).

For numerical solution of most boundary value problems, the finite difference method is the most commonly used algorithm, which could be difficult to use for some cases due to the loss of accuracy or important properties of the original equation, for example, the conservation property.

To derive the precise time integration method for the Riccati equation, we first need to establish the equations that connect the state vectors $\mathbf{q}_{\mathrm{a}}, \mathbf{p}_{\mathrm{a}}$ at $t = t_{\mathrm{a}}$, with $\mathbf{q}_{\mathrm{b}}, \mathbf{p}_{\mathrm{b}}$ at $t = t_{\mathrm{b}}$. If the interval $(t_{\mathrm{a}}, t_{\mathrm{b}})$ is considered as an interval of the entire integration domain $[0, t_{\mathrm{f}}]$, the equations can be expressed as

$$\mathbf{q}_{\mathrm{b}} = \mathbf{F}\mathbf{q}_{\mathrm{a}} - \mathbf{G}\mathbf{p}_{\mathrm{b}} \tag{7a}$$

$$\mathbf{p}_{\mathrm{a}} = \mathbf{Q}\mathbf{q}_{\mathrm{a}} + \mathbf{E}\mathbf{p}_{\mathrm{b}} \tag{7b}$$

where $\mathbf{F}, \mathbf{G}, \mathbf{Q}, \mathbf{E}$ are $n \times n$, $n \times m$, $m \times n$, $m \times m$ matrices, respectively, to be determined. For time independent system, the matrices $\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{D}$ are independent of t. Hence, the $\mathbf{F}, \mathbf{G}, \mathbf{Q}$ and $\mathbf{E}$ will only depend on the length of the interval

$$\Delta t = t_{\mathrm{b}} - t_{\mathrm{a}} \tag{8}$$

Treating $\mathbf{q}_{\mathrm{a}}, \mathbf{p}_{\mathrm{a}}$ as the given initial vectors at t_{a}, and taking the partial derivative of equation (7a) with respect to t_{b}, we have

$$\frac{\partial \mathbf{q}_{\mathrm{b}}}{\partial t_{\mathrm{b}}} = \frac{\partial \mathbf{F}}{\partial t_{\mathrm{b}}} - \frac{\partial \mathbf{G}}{\partial t_{\mathrm{b}}} - \mathbf{G}\frac{\partial \mathbf{p}_{\mathrm{b}}}{\partial t_{\mathrm{b}}}, \qquad \mathbf{0} = \frac{\partial \mathbf{Q}}{\partial t_{\mathrm{b}}} + \frac{\partial \mathbf{E}}{\partial t_{\mathrm{b}}} + \mathbf{E}\frac{\partial \mathbf{p}_{\mathrm{b}}}{\partial t_{\mathrm{b}}}, \tag{9a}$$

Since equation (5) can be written as

$$\frac{\partial \mathbf{q}_b}{\partial t_b} = \mathbf{A}\mathbf{q}_a + \mathbf{D}\mathbf{p}_b, \qquad \frac{\partial \mathbf{p}_b}{\partial t_b} = \mathbf{B}\mathbf{q}_a + \mathbf{C}\mathbf{p}_b. \tag{10a}$$

Equations (10) can be substituted into (9) to get

$$\frac{\partial \mathbf{F}}{\partial t_b} - (\mathbf{GB} + \mathbf{A})\mathbf{q}_b - (\mathbf{D} + \mathbf{GC} + \frac{\partial \mathbf{G}}{\partial t_b})\mathbf{p}_b = \mathbf{0}, \tag{11a}$$

$$\frac{\partial \mathbf{Q}}{\partial t_b}\mathbf{q}_a + \mathbf{EB}\mathbf{q}_b + (\mathbf{EC} + \frac{\partial \mathbf{E}}{\partial t_b})\mathbf{p}_b = \mathbf{0} \quad . \tag{11b}$$

Note further that the vectors $\mathbf{q}_a, \mathbf{q}_b$ and $\mathbf{p}_b$ in equation (11) are not linearly independent. Substituting equation (7a) into (11), we obtain

$$\left[\frac{\partial \mathbf{F}}{\partial t_b} - (\mathbf{GB} + \mathbf{A})\mathbf{F}\right]\mathbf{q}_a + \left[\mathbf{AG} + \mathbf{GBG} - \mathbf{D} - \mathbf{GC} - \frac{\partial \mathbf{G}}{\partial t_b}\right]\mathbf{p}_b = \mathbf{0}, \tag{12a}$$

$$\left[\frac{\partial \mathbf{Q}}{\partial t_b} + \mathbf{EBF}\right]\mathbf{q}_a + \left[\frac{\partial \mathbf{E}}{\partial t_b} + \mathbf{E}(\mathbf{C} - \mathbf{BG})\right]\mathbf{p}_b = \mathbf{0}. \tag{12b}$$

Since $\mathbf{q}_a$ and $\mathbf{p}_b$ are linearly independent, equation (12) leads to the following

$$\frac{\partial \mathbf{G}}{\partial t_b} = \mathbf{AG} + \mathbf{GBG} - \mathbf{D} - \mathbf{GC}, \quad \frac{\partial \mathbf{F}}{\partial t_b} = (\mathbf{GB} + \mathbf{A})\mathbf{F}, \tag{13a}$$

$$\frac{\partial \mathbf{E}}{\partial t_b} = \mathbf{E}(\mathbf{BG} - \mathbf{C}), \quad \frac{\partial \mathbf{Q}}{\partial t_b} = -\mathbf{EBF}. \tag{13b}$$

The initial conditions at $t_b = t_a$ are

$$\mathbf{G} = \mathbf{0}, \quad \mathbf{Q} = \mathbf{0}, \quad \mathbf{E} = \mathbf{I}_m, \quad \mathbf{F} = \mathbf{I}_n, \tag{14}$$

where $\mathbf{I}_m$ and $\mathbf{I}_n$ are identity matrices with dimensions m and n, respectively.

Similarly, we can treat t_a as a variable while fixing t_b, which leads to

$$\frac{\partial \mathbf{G}}{\partial t_a} = \mathbf{FDE}, \qquad \frac{\partial \mathbf{F}}{\partial t_a} = -\mathbf{F}(\mathbf{A} + \mathbf{DQ}), \tag{15a}$$

$$\frac{\partial \mathbf{E}}{\partial t_a} = (\mathbf{C} - \mathbf{QD})\mathbf{E}, \quad \frac{\partial \mathbf{Q}}{\partial t_a} = \mathbf{B} - \mathbf{QA} + \mathbf{CQ} - \mathbf{QDQ}. \tag{15b}$$

The initial conditions are similar to (14) at $t_a = t_b$. For time independent system with matrices $\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{D}$ independent of time, the matrices $\mathbf{F}, \mathbf{G}, \mathbf{Q}$ and $\mathbf{E}$ depend only on the length of the interval $\Delta t = t_b - t_a$. Therefore the relations

$$\mathbf{Q}(t_a, t_b) = \mathbf{Q}(\Delta t), \quad \frac{\partial \mathbf{Q}}{\partial t_b} = \frac{\partial \mathbf{Q}}{\partial (\Delta t)}, \quad \frac{\partial \mathbf{Q}}{\partial t_a} = -\frac{\partial \mathbf{Q}}{\partial (\Delta t)}, \tag{16}$$

hold for matrix $\mathbf{Q}$, and similarly also hold for matrices $\mathbf{F},\mathbf{G},\mathbf{E}$. With these relations, equations (13) can be written as

$$\dot{\mathbf{G}} = \mathbf{AG} + \mathbf{GBG} - \mathbf{D} - \mathbf{GC}, \tag{17a}$$

$$\dot{\mathbf{F}} = (\mathbf{GB} + \mathbf{A})\mathbf{F}, \tag{17b}$$

$$\dot{\mathbf{E}} = \mathbf{E}(\mathbf{BG} - \mathbf{C}), \tag{17c}$$

$$\dot{\mathbf{Q}} = -\mathbf{EBF}. \tag{17d}$$

The dot above $\mathbf{F},\mathbf{G},\mathbf{Q}$ and $\mathbf{E}$ now represents derivatives with respect to Δt. Similarly, equation (15) can be written as

$$\dot{\mathbf{G}} = -\mathbf{FDE}, \tag{18a}$$

$$\dot{\mathbf{F}} = \mathbf{F}(\mathbf{A} + \mathbf{DQ}), \tag{18b}$$

$$\dot{\mathbf{E}} = -(\mathbf{C} - \mathbf{QD})\mathbf{E}, \tag{18c}$$

$$\dot{\mathbf{Q}} = -\mathbf{B} + \mathbf{QA} - \mathbf{CQ} + \mathbf{QDQ}. \tag{18d}$$

Although equations (17) appear to be quite different from (18), it can be proved that they are consistent with each other. Note that equation (18d) is the same as equation in (1). If an algorithm can be developed to calculate the matrix $\mathbf{Q}$ in (18d), such that $\mathbf{Q}$ also satisfies the boundary condition (2), then $\mathbf{Q}$ is the solution matrix $\mathbf{S}$ of (1).

3 Interval Combination

Given two contiguous intervals (t_a, t_b) and (t_b, t_c), we can eliminate the interior state vectors $\mathbf{q}_b, \mathbf{p}_b$ at t_b to form a larger combined interval (t_a, t_c), and obtain equations similar to those in (7) that connect state vectors defined at the two ends t_a and t_c, respectively. Mathematically, the equations for the interval (t_a, t_b) are

$$\mathbf{q}_b = \mathbf{F}_1\mathbf{q}_a - \mathbf{G}_1\mathbf{p}_b, \tag{19a}$$

$$\mathbf{p}_a = \mathbf{Q}_1\mathbf{q}_a + \mathbf{E}_1\mathbf{p}_b, \tag{19b}$$

and those for the interval (t_b, t_c) are

$$\mathbf{q}_c = \mathbf{F}_2\mathbf{q}_b - \mathbf{G}_2\mathbf{p}_c, \tag{20a}$$

$$\mathbf{p}_b = \mathbf{Q}_2\mathbf{q}_b + \mathbf{E}_2\mathbf{p}_c. \tag{20b}$$

To eliminate the interior vectors $\mathbf{q}_b, \mathbf{p}_b$, we solve from (19a) and (20b)

$$\mathbf{q}_b = (\mathbf{I}_n + \mathbf{G}_1\mathbf{Q}_2)^{-1}\mathbf{F}_1\mathbf{q}_a - (\mathbf{I}_n + \mathbf{G}_1\mathbf{Q}_2)^{-1}\mathbf{G}_1\mathbf{E}_2\mathbf{p}_c, \tag{21a}$$

$$\mathbf{p}_b = (\mathbf{I}_m + \mathbf{Q}_2\mathbf{G}_1)^{-1}\mathbf{Q}_2\mathbf{F}_1\mathbf{q}_a + (\mathbf{I}_m + \mathbf{Q}_2\mathbf{G}_1)^{-1}\mathbf{E}_2\mathbf{p}_c, \tag{21b}$$

and substituting (21) into (20a) and (19b), respectively. This leads to, after eliminating $\mathbf{q}_b, \mathbf{p}_b$ and combining the intervals (t_a, t_b) and (t_b, t_c), the equations

$$\mathbf{q}_c = \mathbf{F}_c\mathbf{q}_a - \mathbf{G}_c\mathbf{p}_c, \qquad \mathbf{p}_a = \mathbf{Q}_c\mathbf{q}_a + \mathbf{E}_c\mathbf{p}_c, \tag{22}$$

where

$$\mathbf{G}_{\mathrm{c}} = \mathbf{G}_2 + \mathbf{F}_2(\mathbf{I}_n + \mathbf{G}_1\mathbf{Q}_2)^{-1}\mathbf{G}_1\mathbf{E}_2, \tag{23a}$$

$$\mathbf{Q}_{\mathrm{c}} = \mathbf{Q}_1 + \mathbf{E}_1(\mathbf{I}_m + \mathbf{Q}_2\mathbf{G}_1)^{-1}\mathbf{Q}_2\mathbf{F}_1, \tag{23b}$$

$$\mathbf{F}_{\mathrm{c}} = \mathbf{F}_2(\mathbf{I}_n + \mathbf{G}_1\mathbf{Q}_2)^{-1}\mathbf{F}_1, \qquad \mathbf{E}_{\mathrm{c}} = \mathbf{E}_1(\mathbf{I}_m + \mathbf{Q}_2\mathbf{G}_1)^{-1}\mathbf{E}_2. \tag{23c}$$

4 The 2^N Type Algorithm

In structural mechanics, the substructuring technique has been widely used to improve computational efficiency. If there are multiple identical substructures, only one of them needs to be analyzed and the result can be used for all other identical substructures. This technique has been used successfully for the computation of some optimal control problems [9]. In the present paper, we extend this technique to the solution of matrix Riccati differential equations. Note that the equations in (7) describe state vectors at a small interval from t_{a} to t_{b}, which corresponds to a single substructure, while those in (22) connect state vectors defined at t_{a} and t_{c}, which correspond to the combination of two contiguous substructures after elimination of the state vectors at t_{b}. The 2^N type algorithm described in [10] is very efficient for this kind of combination involving a large number of similar substructures.

Let η be a typical time step length of an interval $[t_{\mathrm{a}}, t_{\mathrm{b}}]$ for the integration of the equations. We can further divide it uniformly into 2^N subintervals of length τ. For example, with $N = 20$, the length of a subinterval is

$$\tau = \eta / 2^N = \eta / 1048576. \tag{24}$$

For time independent systems, all equations corresponding to different subintervals are the same. After $N = 20$ combination steps, all 1048576 subintervals would have been combined to generate a equation system like (7). Note that the entire domain of integration runs from 0 to t_{f}, in which the integration can also be done using the 2^N type algorithm to combining all intervals of length η.

The main part of the computation of this 2^N type algorithm is the repeated execution of

$$\mathbf{G}_{\mathrm{c}} = \mathbf{G} + \mathbf{F}(\mathbf{I}_n + \mathbf{G}\mathbf{Q})^{-1}\mathbf{G}\mathbf{E}, \quad \mathbf{Q}_{\mathrm{c}} = \mathbf{Q} + \mathbf{E}(\mathbf{I}_m + \mathbf{Q}\mathbf{G})^{-1}\mathbf{Q}\mathbf{F}, \tag{25a}$$

$$\mathbf{F}_{\mathrm{c}} = \mathbf{F}(\mathbf{I}_n + \mathbf{G}\mathbf{Q})^{-1}\mathbf{F}, \qquad \mathbf{E}_{\mathrm{c}} = \mathbf{E}(\mathbf{I}_m + \mathbf{Q}\mathbf{G})^{-1}\mathbf{E}, \tag{25b}$$

for N times. Each time the calculated matrices $\mathbf{G}_{\mathrm{c}}, \mathbf{Q}_{\mathrm{c}}, \mathbf{F}_{\mathrm{c}}$ and $\mathbf{E}_{\mathrm{c}}$ are put into the right-hand side of (25) to calculate new matrices for the larger combined intervals.

To start the recursive computation given by equations in (25), it is necessary to generate $\mathbf{G}, \mathbf{Q}, \mathbf{F}$ and $\mathbf{E}$ corresponding to the smallest subinterval of length τ defined by (24). These matrices are defined by equations in (17) (or its equivalent equations in (18)), with the initial conditions in (14). Although equation

(17) is non-linear, the power series expansion method can be used to solve them approximately.

Let $\Delta t = \tau$ in equations (17) and (18), and expand $\mathbf{G}, \mathbf{Q}, \mathbf{F}$ and $\mathbf{E}$ as

$$\mathbf{G}(\tau) = \mathbf{g}_1\tau + \mathbf{g}_2\tau^2 + \mathbf{g}_3\tau^3 + \mathbf{g}_4\tau^4, \quad \mathbf{Q}(\tau) = \mathbf{q}_1\tau + \mathbf{q}_2\tau^2 + \mathbf{q}_3\tau^3 + \mathbf{q}_4\tau^4 \tag{26a}$$

$$\mathbf{F}(\tau) = \mathbf{I} + \mathbf{f}_1\tau + \mathbf{f}_2\tau^2 + \mathbf{f}_3\tau^3 + \mathbf{f}_4\tau^4, \quad \mathbf{E}(\tau) = \mathbf{I} + \mathbf{e}_1\tau + \mathbf{e}_2\tau^2 + \mathbf{e}_3\tau^3 + \mathbf{e}_4\tau^4 \tag{26b}$$

Substituting the first equation in (26a) into (17a) and comparing the coefficients of different powers of τ, we have

$$\mathbf{g}_1 = \mathbf{D}, \ \mathbf{g}_2 = (\mathbf{A}\mathbf{g}_1 - \mathbf{g}_1\mathbf{C})/2, \ \mathbf{g}_3 = (\mathbf{A}\mathbf{g}_2 - \mathbf{g}_2\mathbf{C} + \mathbf{g}_1\mathbf{B}\mathbf{g}_1)/3, \\ \mathbf{g}_4 = (\mathbf{A}\mathbf{g}_3 - \mathbf{g}_3\mathbf{C} + \mathbf{g}_2\mathbf{B}\mathbf{g}_1 + \mathbf{g}_1\mathbf{B}\mathbf{g}_2)/4. \tag{27}$$

Applying similar procedures to (17b), (17c) and (17d), we obtain

$$\mathbf{f}_1 = \mathbf{A}, \quad \mathbf{f}_2 = (\mathbf{A}\mathbf{f}_1 - \mathbf{g}_1\mathbf{B})/2, \quad \mathbf{f}_3 = (\mathbf{A}\mathbf{f}_2 - \mathbf{g}_2\mathbf{B} + \mathbf{g}_1\mathbf{B}\mathbf{f}_1)/3, \\ \mathbf{f}_4 = (\mathbf{A}\mathbf{f}_3 + \mathbf{g}_3\mathbf{B} + \mathbf{g}_2\mathbf{B}\mathbf{f}_1 + \mathbf{g}_1\mathbf{B}\mathbf{f}_2)/4, \tag{28}$$

$$\mathbf{e}_1 = -\mathbf{C}, \quad \mathbf{e}_2 = (\mathbf{B}\mathbf{g}_1 - \mathbf{e}_1\mathbf{C})/2, \quad \mathbf{e}_3 = (\mathbf{B}\mathbf{g}_2 - \mathbf{e}_2\mathbf{C} + \mathbf{e}_1\mathbf{B}\mathbf{g}_1)/3, \\ \mathbf{e}_4 = (\mathbf{B}\mathbf{g}_3 - \mathbf{e}_3\mathbf{C} + \mathbf{e}_2\mathbf{B}\mathbf{g}_1 + \mathbf{e}_1\mathbf{B}\mathbf{g}_2)/4, \tag{29}$$

$$\mathbf{q}_1 = -\mathbf{B}, \quad \mathbf{q}_2 = -(\mathbf{B}\mathbf{f}_1 + \mathbf{e}_1\mathbf{B})/2, \quad \mathbf{q}_3 = -(\mathbf{B}\mathbf{f}_2 + \mathbf{e}_2\mathbf{B} + \mathbf{e}_1\mathbf{B}\mathbf{f}_1)/3, \\ \mathbf{q}_4 = -(\mathbf{B}\mathbf{f}_3 + \mathbf{e}_3\mathbf{B} + \mathbf{e}_2\mathbf{B}\mathbf{f}_1 + \mathbf{e}_1\mathbf{B}\mathbf{f}_2)/4. \tag{30}$$

Higher order approximations can be easily obtained in a similar way, but is unnecessary. Substituting the coefficient matrices given by (27)-(30) into equation (26), we obtain approximations of $\mathbf{G}, \mathbf{Q}, \mathbf{F}$ and $\mathbf{E}$ for the subinterval of length $\Delta t = \tau$.

Note that all formulations before equation (26) are exact. There are truncation errors caused by disregarding terms of order higher than four in equation (26). For stiff problems, a larger ***N*** can be used to further reduce the truncation error in (26).

The use of 2^N type algorithm, however, changes the order of integration of equation (17), because the combination of subintervals does not proceed in exactly the same order as from t_f backward to 0. For example, to combine three contiguous subintervals numbered 1-3 into a new subinterval C, we can proceed in two obvious ways: 1) Combine subintervals 1 and 2 to get a new subinterval A, then combine subintervals A and 3 to get the final interval C; 2) Combine subintervals 2 and 3 to get subinterval B, then combine subintervals 1 and B to get the final interval C. Based on the matrix inversion lemma [11] and the combination equations in (23), it is not difficult to prove that the results from both combinations are identical.

For practical implementation, noted that the direct use of the combination equations in (25) would cause serious round-off errors when the length of the subintervals τ is very small. To avoid this, the matrices $\mathbf{F}$ and $\mathbf{E}$ should be written as

$$\mathbf{F} = \mathbf{I}_n + \mathbf{F}', \quad \mathbf{E} = \mathbf{I}_m + \mathbf{E}', \quad \mathbf{F}_c = \mathbf{I}_n + \mathbf{F}'_c, \quad \mathbf{E}_c = \mathbf{I}_m + \mathbf{E}'_c \tag{31}$$

and equation (25) should be replaced by

$$\mathbf{G}_c = \mathbf{G} + (\mathbf{I}_n + \mathbf{F}')(\mathbf{G}^{-1} + \mathbf{Q})^{-1}(\mathbf{I}_n + \mathbf{E}') \tag{32a}$$

$$\mathbf{Q}_{\mathrm{c}} = \mathbf{Q} + (\mathbf{I}_m + \mathbf{E}')(\mathbf{Q}^{-1} + \mathbf{G})^{-1}(\mathbf{I}_m + \mathbf{F}') \tag{32b}$$

$$\mathbf{F}'_{\mathrm{c}} = -(\mathbf{I}_n + \mathbf{F}')[\mathbf{GQ}(\mathbf{I}_n + \mathbf{GQ})^{-1} + (\mathbf{I}_n + \mathbf{GQ})^{-1}\mathbf{GQ}](\mathbf{I}_n + \mathbf{F}')/2 + 2\mathbf{F}' + \mathbf{F}'^2 \tag{32c}$$

$$\mathbf{E}'_{\mathrm{c}} = -(\mathbf{I}_m + \mathbf{E}')[\mathbf{QG}(\mathbf{I}_m + \mathbf{QG})^{-1} + (\mathbf{I}_m + \mathbf{QG})^{-1}\mathbf{QG}](\mathbf{I}_m + \mathbf{E}')/2 + 2\mathbf{E}' + \mathbf{E}'^2 \tag{32d}$$

5 Conservative Systems

For continuous time optimal control and elastic wave propagation problems, the system being studied are conservative. In these cases, we have $m = n$, and the matrices $\mathbf{D}$ and $\mathbf{B}$ in dual equations (5) are symmetric with

$$\mathbf{C} = -\mathbf{A}^{\mathrm{T}}, \quad \mathbf{D} = \mathbf{D}^{\mathrm{T}}, \quad \mathbf{B} = \mathbf{B}^{\mathrm{T}}. \tag{33}$$

Similarly, matrices $\mathbf{Q}$ and $\mathbf{G}$ in equation (7) are also symmetric with

$$\mathbf{F} = \mathbf{E}^{\mathrm{T}}, \quad \mathbf{G} = \mathbf{G}^{\mathrm{T}}, \quad \mathbf{Q} = \mathbf{Q}^{\mathrm{T}}. \tag{34}$$

Actually, (7) is the integrated form of (5) based on the theory of Hamiltonian systems [9]. Substituting (34) into (7), we have

$$\mathbf{q}_{\mathrm{b}} = \mathbf{F}\mathbf{q}_{\mathrm{a}} - \mathbf{G}\mathbf{p}_{\mathrm{b}}, \quad \mathbf{p}_{\mathrm{a}} = \mathbf{Q}\mathbf{q}_{\mathrm{a}} + \mathbf{F}^{\mathrm{T}}\mathbf{p}_{\mathrm{b}}, \tag{35}$$

which can be rewritten as

$$\begin{Bmatrix} \mathbf{q}_{\mathrm{b}} \\ \mathbf{p}_{\mathrm{b}} \end{Bmatrix} = \mathbf{T}\begin{Bmatrix} \mathbf{q}_{\mathrm{a}} \\ \mathbf{p}_{\mathrm{a}} \end{Bmatrix}, \qquad \text{with} \qquad \mathbf{T} = \begin{bmatrix} \mathbf{F} + \mathbf{G}\mathbf{F}^{-\mathrm{T}}\mathbf{Q} & -\mathbf{G}\mathbf{F}^{-\mathrm{T}} \\ -\mathbf{F}^{-\mathrm{T}}\mathbf{Q} & \mathbf{F}^{-\mathrm{T}} \end{bmatrix} \tag{36}$$

It is easy to verify that

$$\mathbf{T}^{\mathrm{T}}\mathbf{J}\mathbf{T} = \mathbf{J}, \quad \text{with} \quad \mathbf{J} = \begin{bmatrix} \mathbf{0} & \mathbf{I} \\ -\mathbf{I} & \mathbf{0} \end{bmatrix}, \tag{37}$$

so $\mathbf{T}$ is a symplectic matrix. Therefore it is easy to find the integration invariant of (35) in the form of

$$\Lambda = \mathbf{v}^{\mathrm{T}}\mathbf{J}\mathbf{P}\mathbf{v}, \qquad \mathbf{v} = \begin{Bmatrix} \mathbf{q} \\ \mathbf{p} \end{Bmatrix}, \tag{38}$$

where $\mathbf{v}$ is the state vector and $\mathbf{P}$ is a $2n \times 2n$ matrix. It is necessary to find the condition for $\mathbf{P}$ so as to keep Λ invariant. It is not difficult to show that if the multiplication of $\mathbf{P}$ and $\mathbf{T}$ is commutative, i.e.

$$\mathbf{P}\mathbf{T} = \mathbf{T}\mathbf{P}, \tag{39}$$

then Λ remains invariant under transformation $\mathbf{T}$. This implies that if $\mathbf{P}$ is any polynomial of $\mathbf{T}$ then Λ is invariant. A good numerical scheme should maintain all invariants in order to correctly represent the behavior of a conservative system.

6 Solution of the Riccati Differential Equation

Based on the discussions in the previous sections, the matrices $\mathbf{G}(\tau)$, $\mathbf{Q}(\tau)$, $\mathbf{F}(\tau)$, and $\mathbf{E}(\tau)$ of the small subinterval of length τ can be computed first by the equations in (26). Then the 2^N type algorithm can be used to calculate these matrices for a typical time interval $[t_a, t_b]$ of length η. Based on these typical interval matrices, the final matrix function $\mathbf{Q}(t)$ can be calculated, which satisfies the differential equation (18d), the same as equation (1). However, the boundary condition that $\mathbf{Q}(t)$ satisfies is given in (14) as

$$\mathbf{Q} = \mathbf{0}, \qquad \text{at } t = t_f, \tag{40}$$

which is not the same as that given in condition (2) for $\mathbf{S}(t)$. On the other hand, the differential equation (17a) for the matrix function $\mathbf{G}(t)$ is the same as equation (3) for the matrix function $\mathbf{T}(t)$, however, the initial condition for $\mathbf{G}(t)$ is (14)

$$\mathbf{G}(0) = \mathbf{0}, \qquad \text{at } t = 0, \tag{41}$$

which is again different from the condition in (4) for $\mathbf{T}(t)$.

To satisfy the boundary condition (2), we need to construct the matrix function $\mathbf{S}(t)$ from the functions $\mathbf{G}(t), \mathbf{Q}(t), \mathbf{F}(t)$ and $\mathbf{E}(t)$ by the equation

$$\mathbf{S}(t) = \mathbf{Q} + \mathbf{E}(\mathbf{I}_m + \mathbf{S}_f\mathbf{G})^{-1}\mathbf{S}_f\mathbf{F}. \tag{42}$$

Since $\mathbf{E} \to \mathbf{I}_m$, $\mathbf{F} \to \mathbf{I}_n$, $\mathbf{G} \to \mathbf{0}$ and $\mathbf{Q} \to \mathbf{0}$ as $t \to t_f$, it can be easily verified that the $\mathbf{S}(t)$ given in equation (42) satisfy the boundary condition given in (2). To show that $\mathbf{S}(t)$ in (42) satisfies equation (1), we need to use the relation $\mathrm{d}\mathbf{X}^{-1}/dt = -\mathbf{X}^{-1}\dot{\mathbf{X}}\mathbf{X}^{-1}$ and equations (18). The physical interpretation for the above equation is the use of combination equation (23) for the interval (t, t_f) with matrices $[\mathbf{G}, \mathbf{Q}, \mathbf{F}, \mathbf{E}]$ being treated as interval 1, and at the end $t = t_f$ a fictitious interval with matrices $[\mathbf{0}, \mathbf{S}_f, \mathbf{I}_n, \mathbf{I}_m]$ being treated as interval 2. Here, only the equation (23b) is used to obtain equation (42).

The matrix function $\mathbf{T}(t)$ can be constructed similarly by

$$\mathbf{T} = \mathbf{G} + \mathbf{F}(\mathbf{I}_n + \mathbf{G}_0\mathbf{Q})^{-1}\mathbf{G}_0\mathbf{E}. \tag{43}$$

Since $\mathbf{E} \to \mathbf{I}_m$, $\mathbf{F} \to \mathbf{I}_n$, $\mathbf{G} \to \mathbf{0}$ and $\mathbf{Q} \to \mathbf{0}$ as $t \to 0$, it can be easily verified that $\mathbf{T}(t)$ in (43) satisfies the initial condition given in (4). The verification that $\mathbf{T}(t)$ in (43) satisfies differential equation (3) can be done similarly as for $\mathbf{S}(t)$, except that the equations in (17) should be used. Let $\mathbf{S}(t) = \mathbf{S}_\infty$ when $t \to \infty$, we have

$$-\mathbf{B} + \mathbf{S}_\infty\mathbf{A} - \mathbf{C}\mathbf{S}_\infty + \mathbf{S}_\infty\mathbf{D}\mathbf{S}_\infty = \mathbf{0}, \tag{44}$$

which is the algebraic Riccati equation. For non-conservative systems, such as the source free transport system and elastic wave propagation with damping, the matrix

$\mathbf{S}_\infty$ can also be calculated using the 2^N type algorithm. In this case, the procedure described in the previous sections should be carried out until $\mathbf{E}$ and $\mathbf{F}$ are nearly zero matrices. The matrices $\mathbf{Q}$ and $\mathbf{G}$ are then $\mathbf{S}_\infty$ and $\mathbf{T}_\infty$, respectively. The algebraic Riccati equation for $\mathbf{T}_\infty$ is

$$-\mathbf{D}-\mathbf{T}_\infty\mathbf{C}+\mathbf{A}\mathbf{T}_\infty+\mathbf{T}_\infty\mathbf{B}\mathbf{T}_\infty=\mathbf{0} \tag{45}$$

7 Numerical Examples

Although the 2^N type precise time integration is applicable to problems with finite integration domain $[0,t_{\mathrm{f}}]$, we choose a infinite domain in these examples to demonstrate its application to algebraic Riccati equation.

Example 1: $n=4;\ \ m=4;$ the system matrices are given as

$$\mathbf{A}=\begin{bmatrix} -0.3379 & 0.5821 & -.1579 & 0.2771 \\ -267825 & -.1705 & 0 & 0 \\ -.11821 & -.3059 & -.5523 & 0.9694 \\ 0 & 0 & 0 & 7.6923 \end{bmatrix},\ \mathbf{B}=\begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & -10-10\mathrm{i} & 0 & 0 \\ 0 & 10 & -100\text{-}100\mathrm{i} & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix},$$

$$\mathbf{C}=\begin{bmatrix} 0.4+0.4\mathrm{i} & 20+10\mathrm{i} & 0.1 & 0 \\ -0.5 & 0.2 & 0.3 & 0 \\ 0.1 & 0 & 0.5 & 0 \\ -.2 & 0 & -0.1 & -7.7 \end{bmatrix},\ \mathbf{D}=\mathrm{diag}\begin{bmatrix} 0 & 0 & 0 & -10.0\text{-}1.0\mathrm{i} \end{bmatrix}.$$

The algebraic Riccati equation (46) was solved by using the 2^N type precise time integration algorithm with $\eta=1.0$ and 4.0, respectively. The calculated matrices $\mathbf{S}_\infty$ are exactly the same. This indicates that the accuracy has reached machine precision, so using a smaller η will not further improve the accuracy. Substituting $\mathbf{S}_\infty$ into (44), we found that the entries in the residual matrix are all smaller than 10^{-10}. Similarly, the calculated matrix $\mathbf{T}_\infty$ also satisfies equation (45) with entries in the residual matrix smaller than 10^{-10}.

Example 2. In this example, we have $n=5;m=1.$ The system matrices are

$$\mathbf{A}=\begin{bmatrix} -0.8 & 0.5 & -0.4 & 0.2 & 0.4 \\ 0.3 & -2.1 & 0 & 0 & 0 \\ 0.1 & 0.3 & -0.5 & 0.2 & 0.6 \\ 0 & 0 & 0 & -0.8 & 0.5 \\ 0.3 & 1.0 & 0 & 0 & -0.9 \end{bmatrix},\ \mathbf{B}=\begin{bmatrix} 0 \\ -5.0 \\ 0 \\ 0 \\ 0 \end{bmatrix}^{\mathrm{T}},\ \mathbf{C}=\begin{bmatrix} 0.5 \end{bmatrix},\ \mathbf{D}=\begin{bmatrix} 0 \\ 0 \\ 0 \\ 2.0 \\ 0 \end{bmatrix}.$$

The algebraic Riccati equations (44) and (45) were solved using $\eta = 0.4$ and $\eta = 5.0$, respectively. The numerical results are exactly the same for both cases. Substitute the matrix $\mathbf{S}_\infty$ and $\mathbf{T}_\infty$ into (44) and (45), respectively, we found again that the entries in the residual matrices are smaller than 10^{-10}.

8 Concluding Remarks

The 2^N type precise time integration algorithm discussed in this paper is very efficient for calculating accurate solutions to matrix Riccati equations. The computer programming for this method is also straightforward since it uses only matrix operations.

References

1. Bellman, R.: Methods of non-linear analysis, vol. 2, Academic Press, NY, 1973.
2. Bittanti, S., Laub, A. J. Willems, J. C.: The Riccati Equation. Springer-Verlag, NY, 1991.
3. Green, M., Limebeer, D. J. N.: Linear Robust control. Prentice-Hall, Englewood Cliff, NJ, 1995.
4. Basar, T., Bernland, P.: H_∞ Optimal Control and Related Mini-Max Design Problems--A dynamic game approach, 2nd Ed. Birkhauser, Boston, 1995.
5. Kenney, C. S., Leipnik, R. B.: Numerical integration of the differential Riccati equation. IEEE Trans, *AC,* 30 (1985) 962.
6. Zhong, W. X., Williams, F. W.: A precise time integration method. Proc. Inst. Mech. Engrs. 208 (1994) 427-430.
7. Zhong, W. X.: Precise integration of eigen-waves for layered-media, in Proc. EPMESC-5. 2 (1995) 1209-1220.
8. W.X. Zhong, 'The method of precise integration of finite strip and wave guide problems', Proc. Intern. Conf. on Computational Method in Struct. and Geotech. Eng. (1994) 50-60.
9. Zhong, W. X., Lin, J. H., Qiu, C. H.: Computational structural mechanics and optimal control---The simulation of substructural chain theory to linear quadratic optimal control problems, Intern. J. Num. Meth. Eng. 33 (1992) 197-211.
10. Angel, E., Bellman, R.: Dynamic Programming and Partial Differential Equations. Academic Press, New York, 1972.
11. Stengel , R. F.: Stochastic Optimal Control. John Wiley and Sons, New York, 1986.

GEA: A Complete, Modular System for Generating Evaluative Arguments

Giuseppe Carenini

Department of Computer Science
University of British Columbia
2366 Main Mall ,Vancouver, B.C. Canada V6T 1Z4
carenini@cs.ubc.ca

Abstract. This paper presents a system for generating user tailored evaluative arguments, known as the Generator of Evaluative Arguments (GEA). GEA design is based on a pipelined architecture commonly used in natural language generation. After an overview description of GEA main components, we focus on how GEA performs the microplanning tasks. Details are provided by examining the generation of a sample argument.

1 Introduction

Evaluative arguments are communicative acts that attempt to advise or persuade the addressee that something is good (vs. bad) or right (vs. wrong). The ability to generate evaluative arguments is critical in many communicative settings involved in human-computer interaction. For instance, a system that serves as a student advisor may need to justify why a particular course is a good choice for its user. Or, in a different context, a real-estate software assistant may need to argue that one house is a terrible choice for its user. In the field of natural language generation (NLG), considerable research has been devoted to develop computational models for automatically generating user tailored evaluative arguments. Among others, [1-3] have investigated the process of selecting and structuring the argument content, while [4] developed a detailed model of how the selected content should be realised into natural language. However, a key limitation of previous research is that specific projects have tended to focus on only one aspect of the generation process, leaving the development of a comprehensive computational model as future work. In this paper, we present a preliminary attempt to develop such a model. By extending and integrating previous work, we have designed and implemented the Generator of Evaluative Arguments (GEA), a complete NLG system that covers all aspects of the generation process. GEA uses (as much as possible) domain-independent data structures and algorithms and encodes general principles on how evaluative arguments are to be generated. In the reminder of this paper, we first present a standard architecture for a generic NLG system. Then, we describe how the modules and tasks of this architecture have been instantiated in GEA to model the generation of evaluative arguments. After that, we

V.N. Alexandrov et al. (Eds.): ICCS 2001, LNCS 2073, pp. 959-968, 2001.
© Springer-Verlag Berlin Heidelberg 2001

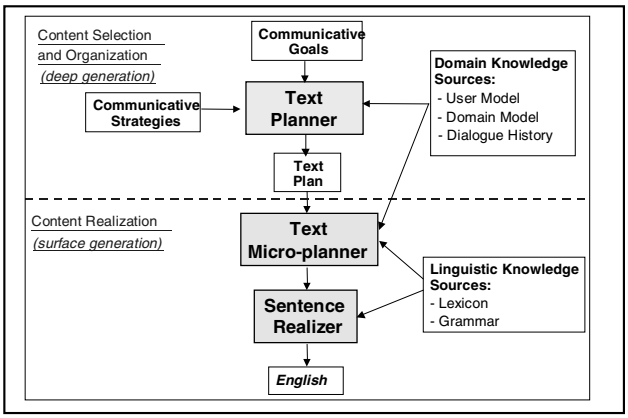


Fig. 1. NLG system pipeline architecture

focus on the GEA text microplanner module. Details are provided by examining the generation of a sample user tailored evaluative argument in the real-estate domain.

2 Standard Architecture for a Generic NLG System

Text generation involves two fundamental tasks: a process that selects and organizes the content of the text (deep generation), and a process that expresses the selected content into natural language (surface generation). Most previous work in NLG makes the assumption that deep generation should strictly precede surface generation. The resulting pipeline architecture, which is adopted in this work, is shown in Fig. 1. In this architecture language generation is modeled as a goal-driven communicative process. The initial input of deep generation is a set of communicative goals, typically of the form: make the hearer *believe* something, change the hearer *attitude* about something and make the hearer *intend to do* something. Then, a *Text Planner* selects and organizes content to achieve the communicative goals given as input (see [5]). In performing this task, the text planner applies a set of communicative strategies that specify for each communicative goal how it can be achieved by either posing further communicative sub-goals for the planner, or by performing a primitive communicative action. The application of these strategies typically relies on three domain knowledge sources (see Fig. 1): a *domain model*, a *user model* and a *dialogue history*. The domain model is the source from which the content of the text is selected, while the user model and the dialogue history allow the planner to tailor the content and structure of the text to both features of the user and features of previous interaction. For instance, consider the communicative goal of increasing the user positive attitude towards an entity (e.g., a house). The text planner would select information about the house and related entities (e.g., the house's neighborhood) from a model of the real-estate domain. Furthermore, the text planner may select different information depending on the user, because different users might agree on the same evaluation for different reasons depending on their preferences. And finally, as an example of sensitivity

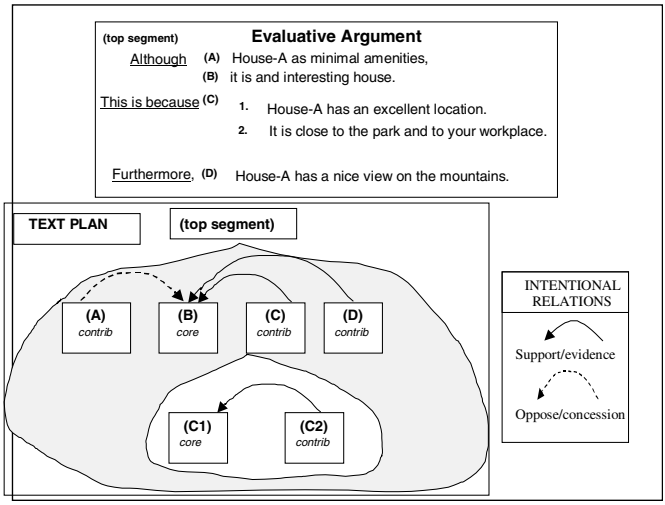


Fig. 2. Sample text plan

to the dialog history, the text planner may avoid repeating information that had been already presented to the hearer in a previous interaction.

The output of text planning is a *text plan*, a data structure that specifies: the rhetorical structure of the text, what propositions the text should convey and a partial order between those propositions. The rhetorical structure of the text subdivides the text into segments. Each segment is a portion of text that is intended to achieve a communicative goal. Segments are internally structured and consist of an element that most directly expresses the segment purpose and any number of constituents supporting that purpose.[1] The rhetorical structure also specifies for each supporting element how it relates to the main element, both from an intentional perspective, (i.e., how the supporting element is intended to support the main one), and from an informational perspective, (i.e., how its content relates to that of the main element). The text for a sample evaluative argument and its corresponding text plan are shown in Fig. 2. The text plan specifies that the text consists of four subsegments. The third segment is itself composite and consist of two subsegments. Only intentional relations are shown because the informational ones are all obvious properties of the house.

Going back to the pipeline architecture shown in Fig. 1, the generation of a text plan ends the process of *deep generation*. The second task involved in text generation, *surface generation*, comprises the two sub-processes of *text microplanning* and *sentence realization*. Notice that, while text planning primarily relies on domain knowledge and sentence realization primarily relies on linguistic knowledge, microplanning tasks consider the interactions between domain knowledge and linguistic knowledge [6]. The following three tasks belong to microplanning: *(a)* **Lexicalization** is the task

[1] The main element and the supporting elements are called differently in different discourse theories. In this paper we use core vs. contributors respectively. These elements can be composite segments themselves.

of selecting words and associated syntactic structures to express semantic information. Usually, lexicalization also includes the selection of cue phrases (e.g., "although", "because", "in fact"), which are words and phrases that mark the relationship between portions of text. Three basic types of lexicalization can be identified (see [6] for more details and examples). In *simple lexicalization*, a proto-phrase template is associated with each possible chunk of semantic information and an information chunk is lexicalized by instantiating its corresponding template. In *simple lexical choice*, several proto-phrase templates are associated with each proposition and its arguments. So, in addition to instantiating a template, we have the problem of selecting the most appropriate one. This selection is typically based on syntactic factors (e.g., the syntactic category in which the proposition has to be expressed), specific features of the particular information chunk and pragmatic factors (e.g., user knowledge and preferences). Finally, in *fine-grained lexicalization*, it is assumed that the chunks of information given as input are expressed in terms of abstract semantic primitives. And this requires additional more sophisticated processing. *(b)* **Aggregation** is the task of packaging semantic information into sentences. Three basic types of aggregation can be identified [6]. In *simple conjunction*, two or more informational elements are combined within a single sentence by using a connective such as *and*. For instance, two informational elements that could be realized independently as (a1)"House B-11 is far from a shopping area" and (a2)"House B-11 is far public transportation" can be combined and realized as the single sentence "(a1) and (a2)". In *conjunction via shared participants*, two or more informational elements sharing argument positions with the same content are combined to produce a surface form where the shared content is realized only once. For instance, the two informational elements aggregated above in a *simple conjunction* could be combined in a *conjunction via shared participants* as "House B-11 is far from a shopping area and public transportation". Finally, in *syntactic embedding*, an informational element that might have been realized as a separate major clause is instead realized as a constituent embedded into some other realized element. For instance, two informational elements that could be realized independently as "House B-11 offers a nice view" and "House B-11 offers a view on the river" can be combined and realized as "House B-11 offers a nice view on the river". *(c)* The **generation of referring expressions** is the task of determining the semantic content of the noun phrases used to refer to the domain entities mentioned in the text plan. This task also includes determining when a pronoun is the most effective referring expression (i.e., **pronominalization** decision) [2].

After microplanning (see Fig. 1), the *Sentence Realizer* completes the generation process. It runs the output of the *Micro-Planner* through a computational grammar of English that produces English text. We do not discuss the realization process here, because GEA uses an off the shelf system as sentence realizer.

---

[2] No details on the task of generating referring expression proper are given here, because they are not needed to understand our system. Pronominalization is typically based on text segmentation and the related notion of local coherence. We describe our pronominalization algorithm in Section 4.

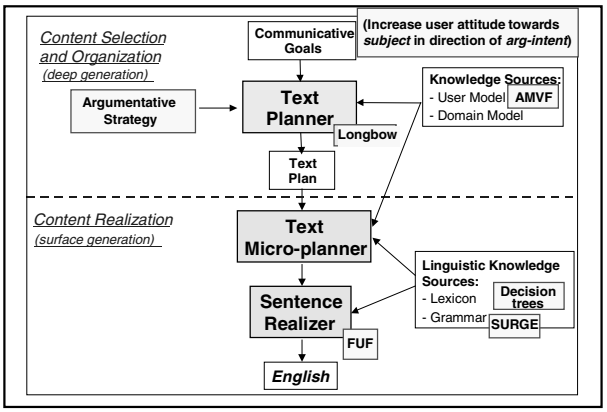


Fig. 3. The GEA architecture

3 The Generator of Evaluative Arguments (GEA)

The design of GEA is based on principles from argumentation theory as well as on previous work in computational linguistics. GEA covers all aspects of generating user tailored evaluative arguments from selecting and organizing the content of the argument, to expressing the selected content into natural language. In this section, we describe the design and development of GEA by illustrating how its architecture specializes the standard architecture of a generic NLG system presented in the previous section (Fig. 1). As shown in Fig. 3, the input to the planning process is an abstract evaluative communicative goal expressing that the user attitude toward a subject should increase in the direction of the communicative intent. In GEA, the subject of the evaluation is an entity in the domain of interest (e.g., a house in the real-estate domain), while the argumentative intent is either positive or negative, with positive/negative meaning that the user should like/dislike the entity.

Given an abstract communicative goal, the Longbow text planner [7] selects and arranges the content of the argument by applying a set of communicative strategies that implement an *argumentation strategy* based on guidelines for content selection and organization from argumentation theory (e.g., [8]). The text planner decomposes abstract communicative goals into primitive ones. In parallel, it also decomposes communicative actions that achieve those goals and imposes appropriate ordering constraints among these actions. Two knowledge sources are involved in this process of goal and action decomposition (see Fig. 3): (i) A *domain model* representing entities and their relationships in a specific domain. (ii) An additive multiattribute value function (*AMVF*), which is a complex model of the user's preferences [9].

An AMVF is a model of a person's values and preferences with respect to entities in a certain class. It comprises a *value tree* and a set of *component value functions*. A value tree is a decomposition of an entity value into a hierarchy of entity aspects (called objectives in decision theory), in which the leaves correspond to the entity primitive objectives (see left of Fig. 4 for a simple value tree in the real estate domain). The arcs in the tree are weighted to represent the importance of an objective

with respect to its siblings (e.g., in Fig. 4 *quality* for UserA is more than twice as important as *amenities* in determining the *house-value*). The sum of the weights at each level is always equal to 1. A component value function for a primitive objective expresses the preferability of each value for that objective as a number in the [0,1] interval, with the most preferable value mapped to 1, and the least preferable one to 0. For instance, in Fig. 4 the *modern* value of the primitive objective *architectural-style* is the most preferred by UserA, and a *distance-from-park* of 1 mile has preferability (1 - (1/3.2 * 1))=0.69. Although for lack of space we cannot provide details here, given a user specific AMVF and an entity, GEA can compute precise quantitative measures that are critical in generating a user-tailored evaluative argument for that entity. First, it is possible to compute how valuable an entity is for that user. Second, GEA can compute how valuable any objective of the entity is for that user (see Fig. 4 on the right for examples). Third, GEA can identify what objectives can be used as supporting or opposing evidence for the evaluation of their parent objective. Fourth, GEA can compute for each objective the strength of supporting (or opposing) evidence it can provide in determining the evaluation of its parent objective. In this way, our argumentation strategy can arrange evidence according to its strength and can generate concise arguments by only including sufficiently strong evidence. Details of the strategy and on the measure of evidence strength are presented in [10]. The argumentation strategy is implemented as a library of plan operators. Given an abstract evaluative communicative goal, the text planner applies the operator library and produces a text plan for an argument intended to achieve that goal.

Next, the text plan is passed to the GEA microplanner which performs aggregation, lexicalization and generates referring expressions. Aggregation, the packaging of semantic information into sentences, is performed according to the standard techniques summarized in Section 2. With respect to lexicalization, the GEA microplanner selects words to express evaluations by following an extension of previous work on realizing evaluative statements [4], whereas decisions about cue phrases (to express discourse relationships among text segments) are implemented as a decision tree based on features suggested in the literature (e.g., [11], [12]). The generation of referring expression in GEA is straightforward; an entity is always referred to by its proper noun. For pronominalization (deciding whether to use a pronoun or not to refer to an entity), simple rules based on centering theory [13] are applied. Finally, the output of text microplanning is unified by the GEA sentence realizer (FUF) with the Systemic Unification Realization Grammar of English (SURGE) [14].

4 The Generation of a Sample Argument

GEA is a complex computer application that integrates and extends several systems and formalisms. For illustration, in this section, we examine the generation of the sample evaluative argument shown in Fig. 6. The argument is about a particular house for a particular user. Information about the house along with an AMVF preference model for the sample user are shown in Fig. 4. For lack of space, we mainly focus here on the key tasks involved in how the GEA's microplanner processes the text plan

for the argument. Details on GEA's argumentation strategy for content selection and organization can be found in [10].

The generation of the argument is initiated by posing the communicative goal `(increased-attitude User-A House-2-33 +)` for the text planner to achieve. By applying the operator library implementing the argumentation strategy, the text planner selects and organizes the content of the argument. This process relies on the user's preference model and on the information about the house (shown in Fig. 4). The selected content (i.e., a subset of the AMVF's objectives and their value for UserA) is organized in a text plan. As described in Section 2, a text plan specifies the rhetorical structure of the text, what propositions the argument should convey and a partial order between those propositions. Fig. 5 shows the text plan generated by GEA for our example. The action decompositions and the relation of *evidence* and *concession* between them express the rhetorical structure. The leaves of the text plan express the propositions the argument should convey (e.g., <`Assert that the` *`Quality`* `of House-2-33 is for UserA 0.82`>). And the nodes of the text plan (i.e., the communicative actions) are ordered (e.g., the action `Assert-opposing-props` should be performed before `Assert-props-in-favor`). Notice that the text plan does not include the objective *Crime* (which is included in the argument). The reason is that the current implementation of the argumentation strategy only processes objectives of depth <3 in the AMVF. The objective *Crime* is reintroduced in the argument by subsequent processing in an ad hoc fashion.

Once the process of content selection is completed, the text plan is passed to the GEA microplanner which performs the following tasks.

Lexicalization proper - The GEA microplanner performs *simple lexical choice* (see Section 2). It selects for each proposition in the text plan the most appropriate proto-phrase to express that proposition. First, the selection is based on the objective of the proposition and then on its value for the current user. For instance, in our sample argument, the proposition `(Location House-2-33 0.6)` according to the portion of the decision tree shown in Fig. 7, is mapped to a proto-phrase which (with pronominalization) is realized as *"it has a reasonable location"*, while the proposition `(Distance-shopping House-2-33 0.84)`, according to the portion of the decision tree for the objective Distance-shopping (not detailed in the figure), is mapped to a proto-phrase which is realized as *"it offers easy access to the shops"*. For lack of precise indications from linguistic theory, the numerical intervals that determine the final decisions are simply based on reasonable estimates.

Aggregation – In general, GEA performs both types of structural aggregation described in Section 2 (i.e., aggregation *via shared participants* and *by syntactic embedding*). To ensure argument coherence, aggregation is only attempted between objectives that are related to a claim by the same rhetorical relation. In our example, we have only one aggregation between the *Location* and *Neighborhood* objectives. The two propositions are aggregated by *syntactic embedding*. The aggregation strategy treats aggregation between *Location* and *Neighborhood* as a special case, because it combines two propositions that are not at the same level in the text plan (the evaluation of the neighborhood is evidence for the evaluation of the location, which in turn is evidence for the value of the house – see plan in Fig. 5).

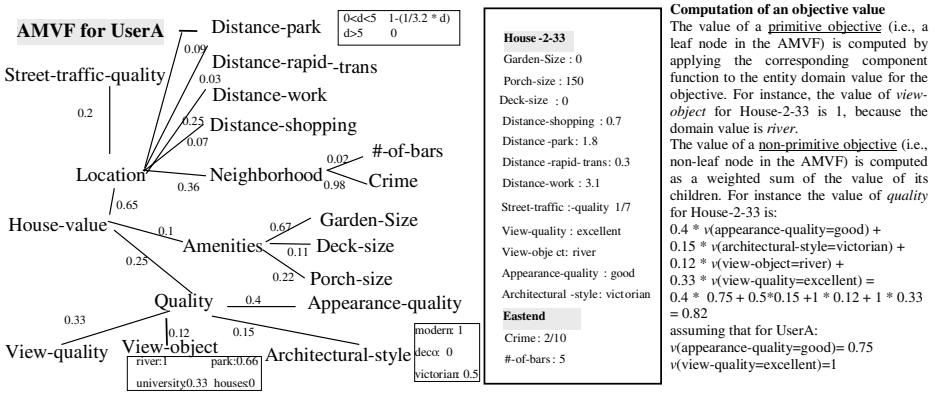


Fig. 4. Preference model for UserA[3], information about House2-33 and value computation

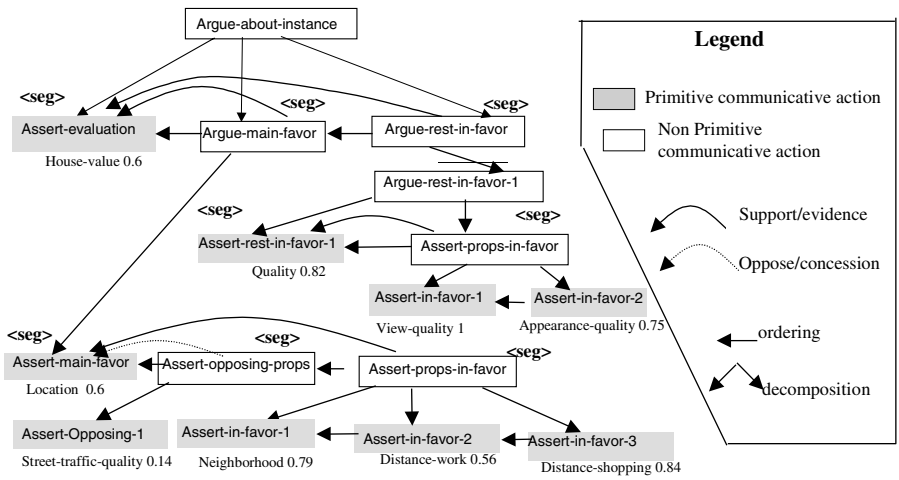


Fig. 5. Text plan and segmentation structure

<<House 2-33 is an interesting house.>^{b} 1<<In fact, it has a reasonable location in the safe Eastend neighborhood. >^{b} 2<Even though the traffic is intense on 2nd street, >^{b} 3<house 2-33 is reasonably close to work. And also it offers an easy access to the shops. >>^{b} 4<<Furthermore, the quality of house 2-33 is good. >^{b} 5<House 2-33 offers an excellent view. And also it looks beautiful. >>>

Fig. 6. Evaluative argument about House-2-33, tailored to UserA

Decision about cue phrases are implemented as a decision tree taking into account relevant features suggested in the literature: (a) the intentional relationship between the core and the contributor (b) the whole segment structure in which core and contributor appear (with core and contributor positions within the segment), and (c) the

[3] For illustration, only three component value functions are shown.

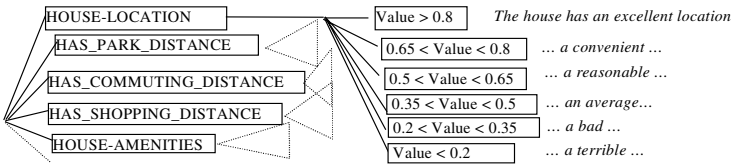


Fig. 7. Decision tree for simple lexical choice in the real-estate domain

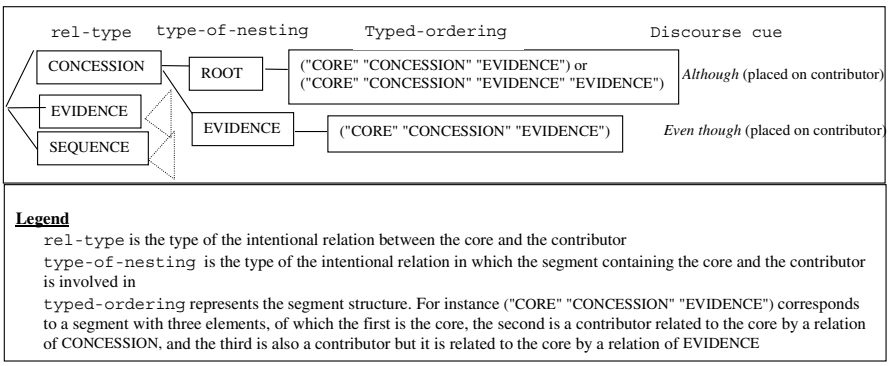


Fig. 8. Portion of decision tree for discourse cue selection

relationship in which the core and contributor segment itself is involved. For illustration, if the reader applies the portion of the decision tree shown in Fig. 8 to the text plan for our example, s/he can verify why "Even though" was used to mark the only concession in our sample argument. For **pronominalization** we have devised simple rules based on centering theory (a theory of local coherence in discourse, see [13]): "In a discourse segment, successive references to the entity evaluated by the argument are realized as pronouns. In contrast, at the beginning of a new segment, the entity is referred to by a pronoun only if two conditions hold. The segment boundary is explicitly marked by a discourse cue and a pronoun has not been used to refer to that entity in the previous sentence". Obviously, applying these rules requires a segmentation of the text given as input. As described in Section 2, the text plan expresses text segmentation: any core or contributor of an intentional rhetorical relation corresponds to a segment. If we apply this definition to our example, we obtain the segment structure shown in Fig. 5 on the text plan and in Fig. 6 on the corresponding text. The text contains five segment boundaries ([b_1… b_5] in Fig. 6). For illustration, the pronominalization rule is applied to the segment boundary b_1 as follows: b_1 is explicitly marked by the "in fact" discourse cue and a pronoun has not been used in the sentence preceding b_1 to refer to `House2-23`. Thus, a pronoun is used to refer to that entity in the following sentence.

5 Conclusions and Future Work

GEA is a fully-implemented, complete and modular NLG system for generating user tailored evaluative arguments. GEA implementation is mostly domain independent.

The system can be easily ported to new domains by simply specifying AMVF models for relevant entities and corresponding decision trees for lexicalization proper. We plan to extend GEA's coverage in at least two ways. First, we intend to enable GEA to generate more complex evaluative arguments (e.g., comparisons between entities). Secondly, we plan to apply GEA to larger AMVFs (i.e., depth => 3). We expect additional techniques to be necessary to generate coherent text for these larger models. Finally, with respect to evaluation, we plan to continue testing GEA following the methodology described in [15] and successfully applied in [16].

References

1. Morik, K., *User Models and Conversational Settings: Modeling the User's Wants*, in *User Models in Dialog Systems*, A.Kobsa and W.Wahlster, Eds. 1989, Springer-Verlag. p. 364-385.
2. Elzer, S., J. Chu-Carroll, and S. Carberry. *Recognizing and Utilizing User Preferences in Collaborative Consultation Dialogues*. in *Proceedings of Fourth International Conference of User Modeling*. 1994. Hyannis, MA.
3. Ardissono, L. and A. Goy. *Tailoring the Interaction with Users in Electronic Shops*. in *Proc. 7th Conference on User Modeling*. 1999. Banff, Canada: Springer-Verlag.
4. Elhadad, M., *Using argumentation in text generation.* Journal of Pragmatics, 1995. **24**: p. 189-220.
5. Moore, J.D., *Participating in Explanatory Dialogues: Interpreting and Responding to Questions in Context*. 1995, Cambridge, MA: MIT Press.
6. Reiter, E. and R. Dale, *Building Natural Language Generation Systems*. Studies in Natural Language Processing. 2000: Cambridge University Press.
7. Young, R.M. and J.D. Moore. *DPOCL: A Principled Approach to Discourse Planning*. in *Proceedings of the 7th Int. Workshop on Text Generation*. 1994. Montreal, Canada.
8. Mayberry, K.J. and R.E. Golden, *For Argument's Sake: A Guide to Writing Effective Arguments*. 2nd ed. 1996: Harper Collins, College Publisher.
9. Clemen, R.T., *Making Hard Decisions: an introduction to decision analysis*. 1996, Belmont, California: Duxbury Press.
10. Carenini, G. and J. Moore. *A Strategy for Generating Evaluative Arguments*. in *International Conference on Natural Language Generation*. 2000. Mitzpe Ramon, Israel. p.47-54
11. Knott, A. and R. Dale, *Choosing a set of coherence relations for text generation: a data-driven approach*, in *Trends in Natural Language Generation: an Artificial Intelligence Perspective (G. Adorni & M. Zock, eds.)*. 1996, Springer-Verlag: Berlin. p. 47-67.
12. Eugenio, B.D., J. Moore, and M. Paolucci. *Learning Features that Predicts Cue Usage*. in *ACL97*. 1997. Madrid, Spain.
13. Grosz, B.J., A.K. Joshi, and S. Weinstein, *Centering: A Framework for Modelling the Local Coherence of Discourse.* Computational Linguistics, 1995. **21**(2): p. 203-226.
14. Elhadad, M. and J. Robin, *An overview of SURGE: a reusable comprehensive syntactic realization component*, . 1996, Dept of Math and CS, Ben Gurion Univ., Israel.
15. Carenini, G. *A Task-based Framework to Evaluate Evaluative Arguments*. in *International Conference on Natural Language Generation*. 2000. Mitzpe Ramon, Israel. p. 9-16
16. Carenini, G. and J. Moore. *An Empirical Study of the Influence of Argument Conciseness on Argument Effectiveness*. in *Annual Meeting of the Association for Computational Linguistics (ACL)*. 2000. Hong Kong, China. p. 150-157

Argumentation in Explanations to Logical Problems

Armin Fiedler and Helmut Horacek

Universität des Saarlandes, FR Informatik,
Postfach 15 11 50, D-66041 Saarbrücken, Germany
{afiedler|horacek}@cs.uni-sb.de

Abstract. Explaining solutions to logical problems is one of the areas where argumentation in natural language plays a prominent role. One crucial reason for the difficulty in pursuing this issue in a systematic manner when relying on formal inference systems lies in the discrepancy between machine-oriented reasoning and human-adequate argumentation. Aiming at bridging these two divergent views, we present a model for producing human-adequate argumentation from machine-oriented inference structures. Ingredients of our method are techniques to build representations to suitable degrees of abstraction and explicitness, and a module for their interactive and adaptive exploration. The presented techniques are not only relevant for the interactive use of theorem provers, but they also have the potential to support the functionality of dialog-oriented tutorial systems.

1 Introduction

Explaining solutions to logical problems is one of the areas where argumentation in natural language plays a prominent role. One crucial reason for the difficulty in pursuing this issue in a systematic manner when relying on formal inference systems lies in the discrepancy between machine-oriented reasoning and human adequate argumentation. The associated differences, however, do not manifest themselves so much in variations in format and syntax, but more substantially in the way how the underlying information is organized.

Aiming at bridging these two divergent views, we present a model for producing human-adequate argumentation from machine-oriented inference structures. Ingredients of our method are techniques to build representations to suitable degrees of abstraction and explicitness, and a module for their interactive and adaptive exploration. The presented techniques are not only relevant for the interactive use of theorem provers, but they also have the potential to support the functionality of dialog-oriented tutorial systems.

This paper is organized as follows. We first provide some background information about presentation of machine-found proofs in natural language. Then we introduce empirical motivations that substantiate divergent demands for human adequate presentations. We describe techniques for building representations meeting these psychological requirements, followed by selection options aiming at summarizations. Then we describe a module for interactive and adaptive exploration. Finally, we illustrate our approach by a moderately complex example.

V.N. Alexandrov et al. (Eds.): ICCS 2001, LNCS 2073, pp. 969–978, 2001.
© Springer-Verlag Berlin Heidelberg 2001

2 Background

The problem of obtaining a natural language proof from a machine-found proof can be divided into two subproblems: First, the proof is transformed from its original machine-oriented formalism into a human-oriented calculus, which is much better suited for presentation. Second, the transformed proof is verbalized in natural language.

Since the lines of reasoning in machine-oriented calculi are often unnatural and obscure, algorithms (see, e.g., [1,14]) have been developed to transform machine-found proofs into more natural formalisms, such as the *natural deduction (ND) calculus* [7]. ND inference steps consist of a small set of simple reasoning patterns, such as forall-elimination ($\forall x P(x) \Rightarrow P(a)$) and implication elimination, that is, modus ponens. However, the obtained ND proofs often are very large and too involved in comparison to the original proof. Moreover, an inference step merely consists of the syntactic manipulation of a quantifier or a connective. [11] gives an algorithm to abstract an ND proof to an *assertion level* proof, where a proof step may be justified either by an ND inference rule or by the application of an assertion (i.e., a definition, axiom, lemma or theorem).

One of the earliest proof presentation systems was [2]. Several theorem provers have presentations components that output proofs in pseudo-natural language using canned text (e.g., [3,4]). Employing several isolated strategies, [5] was the first system to acknowledge the need for higher levels of abstraction when explaining proofs. *PROVERB* [12] expresses machine-found proofs abstracted to the assertion level and applies linguistically motivated techniques for text planning, generating referring expressions, and aggregation of propositions with common elements. Drawing on *PROVERB*, we are currently developing the interactive proof explanation system *P.rex* [6], which additionally features user adaptivity and dialog facilities. [9] is another recently developed NLG system that is used as a back end for a theorem prover.

In order to produce reasonable proof presentations, many systems describe some complex inference steps very densely, and they leave certain classes of proof steps implicit in their output, for example, by abstracting from intermediate inference steps that are recoverable from inductive definitions, or by omitting instantiations of axioms. However, leaving out information on the basis of purely *syntactic* criteria, as this has been done so far, easily leads to incoherent and hardly understandable text portions. In order to get control over the inferability and comprehensibility in presenting inference steps, an explicit model is required which incorporates semantic and pragmatic aspects of communication, which is what we try to achieve by our approach.

3 Empirical Motivation

Issues in presenting deductive proofs, as a special case of presenting argumentative discourse, have attracted a lot of attention in the fields of psychology, linguistics, and computer science. Central insights relevant to deductive argumentation are:

(1) "Let ϱ be an equivalence relation. Therefore we have ϱ is reflexive, we have ϱ is symmetric, and we have ϱ is transitive. Then we have ϱ is symmetric and we have ϱ is reflexive. Then $\forall x : x \varrho x$. Thus we have $h_0 y_0 \varrho h_0 y_0 \ldots$ "
(1') "Let ϱ be an equivalence relation. Thus we have $h_0 y_0 \varrho h_0 y_0 \ldots$"
(2) "Let ϱ be a transitive relation and let $\neg(a \varrho b)$. Let us assume that $c \varrho b$. Hence we have $\neg(a \varrho c)$."
(2') "Let ϱ be a transitive relation and let $\neg(a \varrho b)$. Let us assume that $c \varrho b$. Since ϱ is transitive, $\neg(a \varrho b)$ implies that $\neg(a \varrho c)$ or $\neg(c \varrho b)$ holds. Since we have $\neg(a \varrho b)$ and $c \varrho b$, $\neg(a \varrho c)$ follows."

Fig. 1. Straightforwardly presented proof portions and suitable improvements.

- Logical consequences of certain kinds of information are preferably conveyed implicitly through exploiting the discourse context and default expectations.
- Human performance in comprehending deductive syllogisms varies significantly from one syllogism to another.

The study in [17] demonstrates that humans easily uncover missing pieces of information left implicit in discourse, most notably in sequences of events, provided this information conforms to their expectations in the given context. Similarly to the expectations examined in that study, which occur frequently in everyday conversations, a number of elementary and very common inferences are typically left implicit in mathematical texts, too, including straightforward instantiations, generalizations, and associations justified by domain knowledge.

Another presentation aspect is addressed by studies on human comprehension of deductive syllogisms (see the summary in [13]). These studies have unveiled considerable performance differences among individual syllogisms (in one experiment, subjects made 91% correct conclusions for modus ponens, 64% for modus tollens, 48% for affirmative disjunction, and 30% for negative disjunction). The consequences of this result are demonstrated by the elaborate essay in [18], which presents a number of hypotheses about the impacts that human resource limits in attentional capacity and in inferential capacity have on dialog strategies. These hypotheses are acquired from extensive empirical analysis of naturally occurring dialogs and, to a certain extent, statistically confirmed. One that is of central importance for our investigations says that an increasing number of logically redundant assertions to make an inference explicit are made, in dependency of how hard and important an inference is (modus tollens being an example for a hard inference which requires a more detailed illustration).

In the following, we demonstrate that these crucial issues in presenting deductive reasoning are insufficiently captured by current techniques. Consider the portions of straightforwardly presented proofs produced by an earlier version of *PROVERB*, (texts (1) and (2) in Fig. 1), each of which can be improved significantly, as demonstrated by texts (1') and (2'), correspondingly. Text (1) should be presented more concisely, while parts of text (2) require more explanation. In (1'), the addressees' knowledge about definitions (here, concerning equivalence relations), and their capabilities to mentally perform some sort of simple infer-

ence steps such as conjunction eliminations and elementary substitutions are exploited. In (2'), the involved application of the transitivity axiom is exposed more explicitly through separating the descriptions of the instantiation of the theorem (in reversed direction as a modus tollens) from the disjunction elimination inference, thereby reintroducing the facts not mentioned in immediately preceding utterance parts. Altogether, these examples show some crucial deficits in current proof presentation techniques:

- A large number of easily inferable inference steps is expressed explicitly.
- Involved inferences, though hard to understand, are presented in single shots.

The first deficit suggests the omission of contextually inferable elements in the proof graph, and the second demands the expansion of compound inference steps into simpler parts.

4 Content Determination

In order to obtain presentations similar to (1'), and (2') in Fig. 1, we propose the application of an optimization process that enhances an automatically generated proof at the assertion level. Through this process, pragmatically motivated expansions, omissions, and short-cuts are introduced, and the audience is assumed to be able to mentally reconstruct the details omitted with reasonable effort. In a nutshell, the modified proof graph is built through two subprocesses:

- Building expansions
  Compound assertion level steps are expanded into elementary applications of deductive syllogisms, while marking the original larger steps as summaries.
- Introducing omissions and short-cuts
  Shorter lines of reasoning are introduced by skipping individual reasoning steps, through omitting justifications (marked as inferable) and intermediate reasoning steps (marking the 'indirect' justifications as short-cuts).

In the following, we explain these subprocesses in more detail.

4.1 Level of Abstraction

The purpose underlying the expansion of assertion level steps is to decompose presentations of complex theorem applications or involved applications of standard theorems into easier comprehensible pieces. This operation is motivated by performance difficulties humans typically have in comparable discourse situations. At first, assertion level steps are completely expanded to the natural deduction (ND) level according to the method described in [11]. Thereafter, a partial recomposition of ND steps into inference steps encapsulating the harder comprehensible deductive syllogisms, modus tollens and disjunction elimination steps, is performed, in case the sequence of ND rules in the entire assertion level step contains more than one of these. To do this, the sequence of ND rules is

$$(1)\ \frac{\forall x,y,z:((x\varrho y \wedge y\varrho z) \Rightarrow x\varrho z) \quad \neg(a\varrho c) \quad b\varrho c}{\neg(a\varrho b)}\text{Assertion}$$

$$(2)\ \frac{\dfrac{\dfrac{\forall x,y,z:((x\varrho y \wedge y\varrho z) \Rightarrow x\varrho z)}{(a\varrho b \wedge b\varrho c) \Rightarrow a\varrho c}\forall E \qquad \neg(a\varrho c)}{\dfrac{\neg(a\varrho b \wedge b\varrho c)}{\neg(a\varrho b) \vee \neg(b\varrho c)}\text{NR} \qquad b\varrho c}\Rightarrow E}{\neg(a\varrho b)}\vee E$$

$$(3)\ \frac{\dfrac{\forall x,y,z:((x\varrho y \wedge y\varrho z) \Rightarrow x\varrho z) \quad \neg(a\varrho c)}{\neg(a\varrho b) \vee \neg(b\varrho c)}\text{Modus Tollens} \qquad b\varrho c}{\neg(a\varrho b)}\vee E$$

Fig. 2. An involved assertion level inference at several degrees of abstraction.

broken after each but the last occurrence of a modus tollens or disjunction elimination, and the resulting subsequences of ND steps are composed into a sequence of reasoning steps at some sort of *partial assertion* level. This sequence is then inserted in the proof graph as a potential substitute for the original assertion level step, which is marked as a *summary*. An example for such an expansion and partial recomposition is shown in Fig. 2 ($\forall E$, $\Rightarrow E$, $\vee E$, and NR stand for the ND rules, forall-elimination, implication elimination, disjunction elimination, and natural rewrite, respectively). If $b\varrho c$ and $\neg(a\varrho c)$ hold for a transitive relation ϱ, $\neg(a\varrho b)$ is derivable by a single assertion level step ((1) in Fig. 2). Through expansion to the ND level ((2) in Fig. 2) and recomposition encompassing deductive syllogisms ((3) in Fig. 2), the modus tollens inference step "$\neg(a\varrho c)$ implies $\neg(a\varrho b)$ or $\neg(b\varrho c)$" is separated from the disjunction elimination "Thus, $b\varrho c$ yields $\neg(a\varrho b)$". Note that, in contrast to modus tollens, modus ponens would be composed with disjunction elimination into a single step.

4.2 Degrees of Explicitness

Unlike expanding summaries, creating omissions and short-cuts is driven by communicatively motivated *presentation rules*. They express aspects of human reasoning capabilities with regard to contextually motivated inferability of pieces of information on the basis of explicitly mentioned facts and relevant background knowledge [8]. These rules provide an interface to stored assumptions about the intended audience. They describe the following sorts of situations:

Cut-prop: omission of a *proposition* (premise) appearing as a reason
Cut-rule: omission of a *rule* (axiom instance) appearing as a method
Compactification: short-cut by omitting an *intermediate* inference step

These *reduction* rules aim at omitting parts of a justification that the audience is considered to be able to infer from the remaining justification components of the same line of the proof, or even at omitting an entire assertion level step that is considered inferable from the adjacent inference steps. In order for these

rules to apply successfully, presentation preferences and conditions about the addressees' knowledge and inferential capabilities are checked.

The functionality of the reduction rules can be explained by a simple example. If trivial facts, such as $0 < 1$, or axioms assumed to be known to the audience, such as *transitivity*, appear in the set of justifications of some inference step, they are marked as *inferable* ($0 < 1$ through *Cut-prop*, and *transitivity* through *Cut-rule*). Consequently, the derivation of $0 < a$ can simply be explained by $1 < a$ to an informed audience. Moreover, single facts appearing as the only non-inferable reason are candidates for being omitted through applying *Compactification*. If, for instance, $0 < a$ is the only non-inferable reason of $0 \neq a$, and $0 < a$, in turn, has only one non-inferable reason, $1 < a$, the coherence maintaining similarity between $0 < a$ and $1 < a$ permits omitting $0 < a$ in the argumentative chain. Altogether, $0 \neq a$ can be explained concisely by $1 < a$ to an informed audience.

The presentation rules are matched against proof lines by traversing the proof graph from its leaf nodes and successively continuing to the root node, without back-tracking. In doing this, *Cut-prop* and *Cut-rule* mark locally inferable justification components, and *Compactification* adds alternative justifications through short-cuts.

5 Generating Condensations

In order to convey the information specified completely in view of the assumptions made about the audience summaries are avoided and inferables omitted. Depending on the target item, giving such an explanation in all details required for full understanding may result in a long text. Therefore, it is better to present a reduced first-shot contribution, which can be further investigated interactively, according to user reactions. All possible reductions amount to relaxing the degree of completeness in which the information is presented. In accordance with the aspects of variations focused on, there are two kinds of condensations for obtaining higher degrees of abstraction:

- A sequence of inferences is abstracted into a set of propositions consisting of its conclusion and its premises, while the method how the conclusion is obtained, that is, the underlying sequence of inferences, is omitted. If there is evidence that some of the premises are more important or of more interest to the audience than the remaining ones, larger sets of premises can be reduced to subsets of these. In particular, this measure comprises preferring summaries over detailed exposition of involved inference steps.
- Moreover, in case these inferences constitute the expansion of a pre-designed proof method [15], which underlies the construction of a partial proof, the functionality of that method can be expressed by a descriptive phrase.

Four alternatives are examined, in ascending order of information reduction:

1. Omitting the way how a piece of knowledge (a domain regularity) is applied.
2. Omitting that piece of knowledge.
3. Omitting premises of the inference (eventually, only some of them).

4. Omitting intermediate inference steps.

The choice among these options is based on assumptions about the audience and on the resulting balance of textual descriptions. In [10] we have defined and motivated some strategies for that.

6 Interactive Exploration

Automatically found proofs can be explored interactively with the system *P.rex*. It allows for three types of user interaction: A *command* tells the system to fulfill a certain task, such as explaining a proof. An *interruption* interrupts the system to inform it that an explanation is not satisfactory or that the user wants to insert a different task. In clarification dialogs, finally, the user is prompted to give *answers* to questions that *P.rex* asks when it cannot identify a unique task to fulfill. In this paper, we concentrate on interruptions.

The user can interrupt *P.rex* anytime to enter a new command or to complain about the current explanation. The following speech acts are examples for messages that can be used to interrupt the system:

`(too-detailed :Conclusion` *C*`)`
: The explanation of the step leading to *C* is too detailed, that is, the step should be explained at a more abstract level.

`(too-abstract :Conclusion` *C*`)`
: The explanation of the step leading to *C* is too abstract, that is, the step should be explained at in more detail.

`(too-implicit :Conclusion` *C*`)`
: The explanation of the step leading to *C* is too implicit, that is, the step should be explained more explicitly.

`(too-difficult :Conclusion` *C*`)`
: The explanation of the step leading to *C* is too difficult.

In *P.rex*, `too-difficult` is considered as an underspecified interruption. When the user complains that the derivation of a conclusion *C* was too difficult, the dialog planner enters a clarification dialog to find out which part of the explanation failed to remedy this failure. During the clarification dialog, the system tries to distill whether the user failed to follow some implicit references or whether the explanation was to abstract. The control of the behavior of the dialog planner is displayed in Fig. 3.

7 An Example

We demonstrate the functionality of our model by the presentation of a well-known proof, Schubert's Steamroller [16]:

Axioms:

(1) Wolves, foxes, birds, caterpillars, and snails are animals, and there are some of each of them. Also there are some grains, and grains are plants.

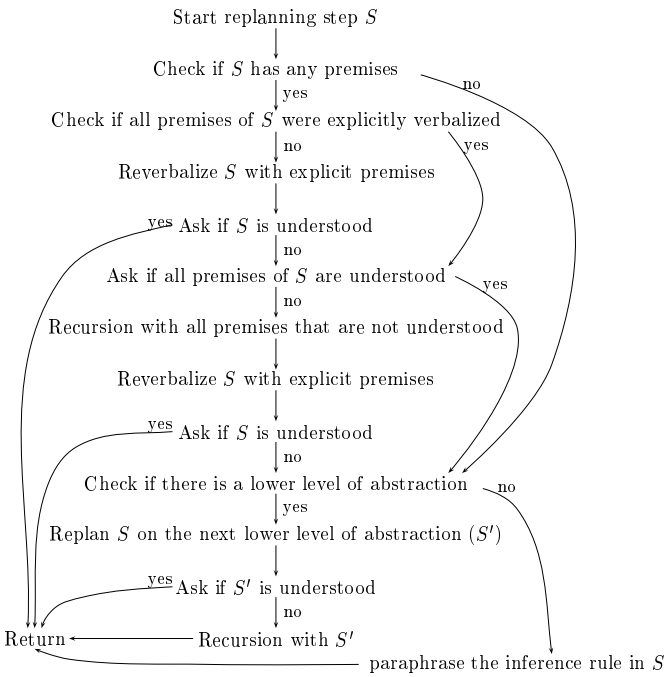


Fig. 3. The reaction of the dialog planner if a step S was too difficult.

(2) Every animal either likes to eat all plants or all animals much smaller than itself that like to eat some plants.
(3) Caterpillars and snails are much smaller than birds, which are much smaller than foxes, which in turn are much smaller than wolves. Wolves do not like to eat foxes or grains, while birds like to eat caterpillars, but not snails. Caterpillars and snails like to eat some plants.

Theorem:

(4) Therefore there is an animal that likes to eat a grain-eating animal.

Proving that theorem (4) is based on applying given pieces of simplified real world knowledge (1) to (3).

In a nutshell, the proof runs along the following lines: Through applying axiom (2) three times, it is first derived that birds eat plants, then that foxes do not eat grains and, finally, that foxes eat the smaller grain-eating birds, the last being the witness needed to prove theorem (4).

Within the theorem proving community, the steamroller problem is famous, because solving it requires several variables to be instantiated purposefully without having a guidance how to do this through the formulation of the theorem to be proved — it has only existentially quantified variables in it, but no constants. Until some years ago, automated theorem provers were unable to apply this technique with sufficient degrees of efficiency, so that they were originally unable to

solve this problem. For our purposes, this problem is attractive for completely different reasons: its definition is easily comprehensible without mathematical knowledge, and a full-detailed solution path is sufficiently complex so that exploring it interactively seems to be well motivated.

Apparently, the proof sketch given above is far from being a complete and fully comprehensible explanation of the proof, since many details that are necessary to understand how the central axiom is applied in each case are not mentioned. The task of the presentation module is then to suitably mediate between such a concise proof sketch and a fully expanded proof description:

System: The proof runs through applying three times the rule that animals either eat plants or all plant eaters smaller than themselves. It is first derived that foxes do not eat grain. Similarly, it is derived that birds eat grain, and finally, that foxes eat birds. Since foxes eat birds, an animal is known that eats a grain-eating animal, q.e.d.
User: More detail, please.
System: The fact that foxes do not eat grain follows from the assumptions that wolves do not eat grain and foxes are smaller than wolves, because animals who do not eat plants eat plant eaters smaller than themselves. Thus, either foxes do not eat grain or they are not smaller than wolves must hold. Hence, only the first alternative is valid. ...
User: Less detail, please.
System: Birds do not eat plant eaters because it is assumed that they do not eat snails, but snails are smaller than birds and they eat plants. And foxes eat birds, because they are plant eaters, and smaller than foxes.
User: Why do birds not eat plant eaters?
System: For animals, it holds that (1) they eat all plants or (2) they eat all plant eaters smaller than themselves. (2) cannot hold for birds, because snails eat plants and they are smaller than birds, but it is known that birds do not eat snails. Hence (1) holds for birds.

8 Conclusion

The analysis of human proof explanations shows that certain logical inferences are only conveyed implicitly drawing on the discourse context and default expectations. Moreover, different syllogisms call for different presentation strategies to account for human performance. In this paper, we proposed the partial assertion level as an appropriate representation of proofs to plan the content of an explanation and different degrees of explicitness and condensation. Then, driven by the unfolding dialog, a reactive planner allows for an interactive, user-adaptive navigation through the proofs.

So far we have implemented *P.rex* and some tools for mediating between levels of abstraction. We are currently investigating manipulations of the proof structure to realize different degrees of explicitness. We will soon incorporate this work into a newly starting project on dialog-oriented tutoring systems. Moreover, we believe that our approach also proves useful for argumentative dialog systems in general.

References

[1] Peter B. Andrews. Transforming matings into natural deduction proofs. In *Proceedings of the 5th International Conference on Automated Deduction*, pages 281–292. Springer Verlag, 1980.

[2] Daniel Chester. The translation of formal proofs into English. *AI*, 7:178–216, 1976.

[3] Yann Coscoy, Gilles Kahn, and Laurent Théry. Extracting text from proofs. In Mariangiola Dezani-Ciancaglini and Gordon Plotkin, editors, *Typed Lambda Calculi and Applications*, number 902 in Lecture Notes in Computer Science, pages 109–123. Springer Verlag, 1995.

[4] Bernd Ingo Dahn, J. Gehne, Thomas Honigmann, and A. Wolf. Integration of automated and interactive theorem proving in ILF. In William McCune, editor, *Proceedings of the 14th Conference on Automated Deduction*, number 1249 in LNAI, pages 57–60, Townsville, Australia, 1997. Springer Verlag.

[5] Andrew Edgar and Francis Jeffry Pelletier. Natural language explanation of natural deduction proofs. In *Proceedings of the 1st Conference of the Pacific Association for Computational Linguistics*, Vancouver, Canada, 1993. Centre for Systems Science, Simon Fraser University.

[6] Armin Fiedler. Using a cognitive architecture to plan dialogs for the adaptive explanation of proofs. In Thomas Dean, editor, *Proceedings of the 16th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 358–363, Stockholm, Sweden, 1999. Morgan Kaufmann.

[7] Gerhard Gentzen. Untersuchungen über das logische Schließen I & II. *Mathematische Zeitschrift*, 39:176–210, 572–595, 1935.

[8] H. Grice. Logic and conversation. *Syntax and Semantics*, 3:43–58, 1975.

[9] Amanda M. Holland-Minkley, Regina Barzilay, and Robert L. Constable. Verbalization of high-level formal proofs. In *Proceedings of the Sixteenth National Conference on Artificial Intelligence (AAAI-99) and Eleventh Innovative Application of Artificial Intelligence Conference (IAAI-99)*, pages 277–284. AAAI Press, 1999.

[10] Hemlut Horacek. Tailoring inference-rich descriptions through making compromises between conflicting cooperation principles. *Int. J. Human-Computer Studies*, 53:1117–1146, 2000.

[11] Xiaorong Huang. Reconstructing proofs at the assertion level. In Alan Bundy, editor, *Proceedings of the 12th Conference on Automated Deduction*, number 814 in LNAI, pages 738–752, Nancy, France, 1994. Springer Verlag.

[12] Xiaorong Huang and Armin Fiedler. Proof verbalization as an application of NLG. In Martha E. Pollack, editor, *Proceedings of the 15th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 965–970, Nagoya, Japan, 1997. Morgan Kaufmann.

[13] Philip Johnson-Laird and Ruth Byrne. *Deduction*. Ablex Publishing, 1990.

[14] Christoph Lingenfelder. *Transformation and Structuring of Computer Generated Proofs*. PhD thesis, Universität Kaiserslautern, Kaiserslautern, Germany, 1990.

[15] Erica Melis. AI-techniques in proof planning. In Henri Prade, editor, *Proceedings of of the 13th European Conference on Artifical Intelligence*, pages 494–498, Brighton, UK, 1998. John Wiley & Sons, Chichester, UK.

[16] Mark E. Stickel. Schubert's steamroller problem: Formulations and solutions. *Journal of Automated Reasoning*, 2:89–101, 1986.

[17] Manfred Thüring and Kurt Wender. Über kausale Inferenzen beim Lesen. *Sprache und Kognition*, 2:76–85, 1985.

[18] Marylin Walker. The effect of resource limitations and task complexity on collaborative planning in dialogue. *Artificial Intelligence*, 85:181–243, 1996.

Analysis of the Argumentative Effect of Evaluative Semantics in Natural Language

Serge V. Gavenko

Moscow State Linguistic University
Usacheva St., 62-510, Moscow 119048 Russia
serge.gavenko@mtu-net.ru

Abstract. The article deals with the semantic aspect of the argumentative discourse. Argumentation has long been studied in the realm of the rhetorical science, though a research into the functioning of linguistic means on which any verbal human interaction is based can present to us some subtle mechanisms that subconsciously govern the process of persuasion and decision-making. The particular research reflected in this paper tries to cast some light upon argumentative power of evaluative semantics. A person who has to make a decision about something, attaches a certain value to it; by building argumentative discourse while employing and operating evaluative notions, authors are able to lead addressees to a certain desired attitude. The highest appeal have sublimated (ethical, aesthetical, emotive) evaluations, though evaluations, based on perceptive sensations may be propelled to have sublimated charge. Such a mechanism is the major point of investigation in the present paper.

1 Argumentation as a Multi-discipline Object

The present research is based on the analysis of evaluative semantics derived from the notion of sense perception used in argumentative discourse. It reflects the development of the study in evaluative semantics of argumentation in the course of my Ph.D. research. This paper will concentrate on a narrower field - that of perceptive evaluations, giving though some insight of the latter's interrelation with other kinds of evaluations.

There are many definitions of argumentation used in various fields of knowledge. A number of theories in various disciplines are involved; these theories will be different according to their object of analysis, their aims, and their methods.

Viewed from a rhetorical point of view argumentation can be defined as a process of proving the validity of some thesis by presenting arguments and counterarguments, evaluating them, and creating in the hearers' minds a belief in the truth and acceptability of the thesis. A distinction has been drawn between logical proof and argumentation; for the former does not concern itself with gaining the agreement of the listener (reader), unlike argumentation, which, for that matter, is a very personal affair, oriented at the audience. An Armenian philosopher and logician, G. Brutyan [3], enlarges the notion of argumentation with the idea

V.N. Alexandrov et al. (Eds.): ICCS 2001, LNCS 2073, pp. 979–988, 2001.
© Springer-Verlag Berlin Heidelberg 2001

that apart from the abovesaid, in argumentation "there are given reasons for accepting the thesis with an aim to work out an active social position and carry out certain programs and actions that follow from it".
The same perspective is employed in the definition by Encyclopedia Britannica: "Argumentation, whether it be called rhetorical or dialectical, always aims at persuading or convincing the audience to whom it is addressed of the value of the theses for which it seeks assent". But there is more to argumentation than the usual logical and rhetoric views, observed for hundreds of years. But it stands to reason that argumentative impact and the success of an act of argumentation depend both on the correctness of the discourse as seen from the point of view of rhetoric laws (the order in which the thesis, arguments and examples are introduced), as well as on the semantics of the linguistic units used in the thesis and arguments. A relatively new approach deals with the question of what argumentation, like all human verbal interaction, is built on, i.e. the language and the laws governing its functioning.
From this point of view, argumentation is a complex of linguistic means used for influencing humans' behavior (i.e. as a factor causing to accept this or that decision) as well as a special type of discourse (alongside description, narration, teaching, analysis, etc.). Such a discourse may be characterized by special communicative and illocutionary aims.
From the point of view of semiotics argumentation may be viewed as a particular sort of communication having a certain effect on the addressee's mind, which is organized by the producer of the speech in accordance with the accepted norms of argumentative discourse. Thus, from the point of view of linguistic structures argumentation is seen as functioning of certain elements of the language; from the point of view of the cognitive approach the linguistic study of argumentation is a part of the investigation into the mental models and communicational models of humans.
A person hearing about an event or a thing in evaluative terms is bound to produce his own attitude to the discussed thing - agreeing with the speaker's assessment or not. If a person accepts a given thesis, he evaluates the latter as having a certain value to him. An opponent must be presented solid evidence in support of the virtues of the discussed point both logically and linguistically, so the person giving arguments may in his turn use some evaluative lexical units to subtly lead the hearer towards the positive assessment in favour of the point being proven. In other words, it is possible to suggest evaluative ideas in the linguistic composition of an act of argumentation.

2 The Role of Evaluative Factor in Argumentation

An argumentative appeal to the evaluative factors is one of most powerful means of argumentation in natural language. Evaluations possess an argumentative charge themselves in the sense that to assess a thing is to make a decision about it. Therefore, the main purpose of evaluation may be viewed as not to report about facts, but to make an influence. An evaluative attitude is in itself a

pragmalinguistic category, which influences the semantic structure of the lexical meaning of words used in argumentative discourse. 'Recommendational' power of evaluation, especially addressing emotiveness, is an argument.

The main purpose of evaluative judgements is not to report about facts but to influence people. In the light of the abovesaid my work argues that one of the most influential means of persuading (and in this sense, persuading subconsciously) is the appeal to assessing things and events from the point of view of evaluations based on notions of sense perception - the initial human way of knowing the world. It should be interesting to analyse the work of evaluative meanings of lexical units which ground their semantic bases in various perceptional sensations.

In other words, we are trying to prove the statement that linguistic evaluations possess argumentative meanings. In the simplest terms this might be presented in the following way: "by default" in most usual situations and pictures of the world pairs like "bright - dark", "warm - cold", etc. represent in their first elements more generally accepted positive notions. Thus, if ideas that possess a positive charge in them are used in describing a point, they are likely to create a positive aura in the attitude to this point in the mind of the recipient. In any case, the subtler the evaluation is, the more likely it is to have an effect of agreement in the listener, proving to him the sound basis for the virtues of the thing in question.

I am grounding my categories of evaluations on the classification by a Russian linguist N.D.Arutyunova [1]. Evaluations - with the degree of subtlety (and argumentative power) ascending - are: perceptive or hedonistic; psychological (intellectual and emotive); aesthetic; ethical; utilitarian; normative; and finally - teleological. Evaluations having a higher level of sublimity appeal to rational and humanitarian standards in people creating thus a powerful means of influence, making people agree with just and positive features of things as described by the author.

My task here is to see how evaluations based on sense perception come to possess features of higher levels - ethical and aesthetical values. In order to understand the way this mechanism works it is necessary to take into consideration three aspects: qualities of the basic meaning of the word; qualities of the new meaning, i.e. various evaluative meanings; and the connection between the basic and the final meanings. In the latter we see the moment of transition from a descriptive field to evaluative one, reconstructing the semantic development which describes the quality in focus at such a transition. In simple terms - why the idea of "coldness" in the assessment of somebody's behavior is ethically less praised than that of "warmness".

According to a well-known theory of Ch. Stevenson [7], language is used by humans descriptively and dynamically. In the realm of the first usage people's knowledge and views are stored, processed, enriched, distributed among people and passed on to other generations. In case of the dynamic usage people give way to feelings, form beliefs and attitudes toward events, and, most importantly for our case, accept motives for performing certain actions.

The same view is held by an English researcher, I.A. Richards [6], who names the functions of the language as referential (a mere statement about an event, when words are used for pointing at the referent) and emotive (used for expressing feelings and attitudes to things and events, as well as making others feel and think the same way). In other schools of linguistics these two functions of language are called illocutional and perlocutional respectively.
The pragmatic aim of argumentative acts belongs to the dynamic (emotive or perloctive) function of the language. As was mentioned, it is understood as making a specific influence on the addressee's mind with the help of linguistic means built up according to the accepted in the given culture argumentative principles. The result which the speaker tries to achieve while carrying out such speech acts is to persuade his opponent by verbal means, influence his choice of picking alternatives in the process of making decisions, and, as a result, conduct his behavior in certain ways.
When speaking about axiological (evaluative) meanings of lexical units it is necessary to mention the fact that besides words carrying direct evaluative meanings, there exist such descriptive lexical units which are not characterized in their dictionary definitions as possessing any kind of evaluation. Nevertheless, despite the initial "descriptiveness" of their usage, the latter may be filled with evaluative meanings depending on the context, which, in its turn, leads to their pragmatic functioning, and in particular, to argumentative meanings.
As a tentative analysis based on such a perception of argumentative discourse I would like to offer the following. For the analysis I chose rather a specific text: it is a piece of forensic discourse. Even though it naturally dictates a certain array of rules governing this very type of speech communication, it still uses English - with the general realia true to this language. My aim here is not to trace the peculiarities of forensic language within the English verbal communication; rather it is to outline the features true to the language in general. In further research I am intending to use materials from various speech styles. This will reveal what similar features are employed by the language as a representation of a particular culture. The analysis does not claim to be reflecting the state of things in the language with total precision; this is the beginning of a search for patterns of evaluative meanings in the mentality of a national culture.

3 A Sample Analysis

The analysis is based on the transcripts of edited and narrated oral court arguments in Roe v. Wade argued on December 13, 1971 in Texas, USA (Edited Supreme Court Opinions Roe v. Wade in [4, pp. 23-45]). Arguments for and against abortion on the territory of the United States of America represent heated debates due to the fact that the subject itself raises highly subjective feelings of both the supporters and antagonists on the issue of abortion, which in its turn is reflected in the character of the speeches.
The aim of this analysis is to try to determine the connectives of evaluative judgements with their perceptive basements in acts of argumentation, and the

role the latter play in influencing recipient's process of decision-making.

A.N. Baranov in his work on the argumentative culture [2] notes that "lexical meanings of many words that are not even oriented at argumentative dialog, at the same time are not at all strange to argumentative meanings". For example Baranov shows that the most usual procedure for introducing a thesis is "employing an explicative modus with verbs of expressing opinion (in the first person singular, present indefinite tense): *suppose, think, consider*, etc. Such words introduce proposition, the truth of which it is necessary to prove". A rhetoric aim here is to show one's awareness of the problem and offer the interlocutors to discuss the truth of the claims. By these means it is also shown that the speaker offers a subjective opinion (i.e. anticipates a possible criticism of his point).

In the analysed text the theses are introduced by verbal constructions in the first person (sg. and pl.) as well as by infinitive phrases with the semantic basis on moduses of visual perception and tactile feelings, and of something which we suggest calling "an inner feeling". The latter is especially true in case of women, which can be viewed as the tendency of women to appeal to intuition (a certain "sixth sense") with further use of emotionally colored evaluations in their acts of argumentation.

When introducing a thesis, people more than often use words dealing with visual perception. Almost all of them are synonymous to the moduses of explicative opinion mentioned by Baranov: *as I see it, as we see it.* Descriptive in their initial meaning, these lexical units introduce the proposition, with an implicit idea that it is a subjective opinion, inviting the opponent to observe it from his own point of view, and (hopefully) accept as true.

Next group of visual moduses concern themselves, on the other hand, with setting objective backgrounds for the offered thesis, they try to point out the real state of things in the world: *in view of the fact that, in the light of*:

In view of all this, we do not agree that, by adopting one theory of life, Texas may override the rights of the pregnant woman that are at stake.

With respect to the State's important and legitimate interest in the health of the mother, the "compelling" point, in the light of present medical knowledge, is at approximately the end of the first trimester.

Therefore, an opponent is forced to make a conclusion based on a self-evident thesis with which he initially is bound to agree, as provided by the speaker. The word apparent has a rather strong argumentative appeal, and it prompts the listener not to doubt the truth of the claim; thus, it is an objective-based evaluation:

The detriment that the State would impose upon the pregnant woman by denying this choice altogether is apparent. Specific and direct harm medically diagnosable even in early pregnancy may be involved.

The word *insight* (*for such insights as the history may afford us*), in the meaning "example" (in-sights) tells us that the review of facts from the past will open to us new perspectives for deciding on the main issue, therefore justifying the necessity of a historical overview and making us pay a special attention to the offered facts:

Before addressing this claim, we feel it desirable to briefly survey, in several aspects, the history of abortion, for such insight as that history may afford us, and then to examine the state purposes and interests behind the criminal abortion laws.

The expression *to color one's thinking and conclusions* has a clearly evaluative motivation behind it, though a certain vagueness as to whether the expression has a negative or positive connotation makes this evaluation ambivalent (the immediate understanding of the phrase can vary from "influence = pressure" to "make picturesque"):

One's philosophy, one's experiences, one's exposure to the raw edges of human existence, one's religious training, one's attitudes towards life and family and their values, and the moral standards one establishes and seeks to observe, are all likely to influence and to color one's thinking and conclusions about abortion.

An adequate understanding of this lexical unit is possible only in the context. In this case the expression reflects the difficulty of the abortion problem and the validity of the reasons given by the supporters on the both sides. Therefore, the speaker calls on the participants to give the problem some serious attention. The same is true of the phrase *vigorous opposite views*:

We forthwith acknowledge our awareness of the sensitive and emotional nature of the abortion controversy, of the vigorous opposite views, even among physicians, and of the deep and seemingly absolute convictions that the subject inspires.

In these two instances the modus of visual perception is a basis for intellectual evaluation (according to the classification by N.D. Arutyunova).
The following quotation may serve as an example of an unsuccessful argumentative act:

Mr. Chief Justice, and may it please the Court. It's an old joke, but when a man argues against two beautiful ladies like this, they are going to have the last word.
(Narrator: No one laughed. Chief Justice Burger looked annoyed. After an embarraced silence, Jay Floyd argued that the case was moot because Jane Roe was no longer pregnant.)

Texas Assistant Attorney General Jay Floyd began his argumentation with a joke that "when a man argues against two beautiful ladies like this, they are going to have the last word". The aesthetic evaluation based on visual perception which is used here is by itself a positive one, but it violated one of Grice's maxims (relevance of the information used), especially being incongruous when used in the court room during important hearings. (This is also shown by the narrator's remarks.)
A.N.Baranov showed that in many cases an appeal to the listener is loaded with some precursory qualities: epistemic (reflecting the speaker's assessment of the possibilities for the alternatives) and axiologic (showing the degree of desirability of this or that answer of the opponent). From this perspective such examples as *as I see it* and *as we see it* may be characterized as epistemic, for they just rep-

resent one of the points of view on this issue, giving the interlocutor freedom to make his own judgements. Other observed examples have axiologic backgrounds because they forward addressees' evaluations on to certain routes.

Further use of evaluative judgements based on visual perception (arguments, counterarguments, examples, conclusions) are grounded on the same principles. For instance, while interpreting The Constitution as having certain rights, it was mentioned that

...these decisions make it clear that not only personal rights that can be deemed "fundamental"... are included in this guarantee of personal privacy. They also make it clear that the right has some extension to activities relating to marriage, ...procreation, ... contraception, ...family relations, ...and child rearing and education.

The repetition of the phrase *make it clear* states an unarguable truth of such information and is the basis for normative (rational) evaluation.

In the course of the pro-choice supporters' arguments it was stated that the laws of Texas are *unconstitutionally vague*, thus giving the basis for negative normative and utilitarian (rational) evaluations:

Roe alleged that she was unmarried and pregnant; that she wished to terminate her pregnancy by an abortion "performed by a competent, licensed physician, under safe, clinical conditions"; that she was unable to get a "legal" abortion in Texas because her life did not appear to be threatened by the continuation of her pregnancy; and that she could not afford to travel to another jurisdiction to in order to secure a legal abortion under safe conditions. She claimed that the Texas statutes were unconstitutionally vague and that they abridged her right of personal privacy, protected by the First, Fourth, Fifth, Ninth, and Fourteenth Amendments.

Truly, a law cannot have such a characteristic, and this remark calls for attention in order to clear things out and make the law understandable.

Same aims as the modus of visual perception are served by haptic (tactile) perception; it can be the basis for intellectual and normative evaluations. When speaking about the difficulties of the discussed issue and about the variety of the points of view, someone used an expression *one's exposure to the raw edges of human existence*, in which the metaphor for the hardships of life is based on the emotionally-utilitarian evaluation expressed by the phrase *raw edges*:

One's philosophy, one's experiences, one's exposure to the raw edges of human existence, one's religious training, one's attitudes towards life and family and their values, and the moral standards one establishes and seeks to observe, are all likely to influence and to color one's thinking and conclusions about abortion.

It was found that in the analysed text emotional evaluations are very often built on tactile perception. As other examples of emotionally colored evaluations the following may serve: aesthetic evaluation in *deformed or defective child*, a repetition of the phrase *she has no relief*, which also carries negative hedonistic evaluation:

If the pregnancy would result in the birth of a deformed or defective child, she has no relief. Regardless of the circumstances of conception, whether it was because of rape, incest, whether she is extremely immature, she has no relief.

and also the verb *to disrupt* as referred to life, body, education, family life and work:

I think it's without question that pregnancy to a woman can completely disrupt her life. It disrupts her body; it disrupts her education; it disrupts her employment; and it often disrupts her entire family life.

All of the examples mentioned above belong to women, and this supports the idea that women have a tendency to influence interlocutors more with the help of emotionally colored arguments rather than merely rational ones. Of course such a conclusion should not follow solely from the presented here relatively insufficient number of examples; but it does serve as proof for conclusions about gender differences in verbal communication drawn by a number of linguists, among whom is D. Tannen [8]. Therefore, the meanings of evaluative nouns also possess a certain situational aspect, which is defined as additional information about conditions and participants of the communication process.

For the speech of males, at least in the analysed text, a very characteristic feature is the use of the so-called war metaphors. It concerns teleologic evaluations based on tactile perception (*important impact, thrust of an attack*), when speaking about the effectiveness of some actions:

The principal thrust of appellant's attack on the Texas statutes is that they improperly invade a right, said to be possessed by the pregnant woman, to chose to terminate her pregnancy.

The original bases for these metaphors (along with the word "invade") are completely obscure now, and for the most part they are widely used without referring to "struggle" or "fight", though the choice of these very phrases has an argumentative aim to make a special impact on the ones who make decisions in this situation.

We called the "inner sense" mentioned above as the basis for a number of evaluations that for the most part (but not always!) characterize the speech of women. Here we include purely emotional evaluations (i.e. emotional by themselves, without context influencing), psychological evaluations and those, based on intuition ("sixth sense"). In the latter category we may include the normative evaluations *"emotional response", "feel that", "feel it desirable"*:

a.
- *Obviously, I have a much more difficult time saying that the state has no interest in late pregnancy.*
- *Why? Why is that?*
- *I think it is more the emotional response to late pregnancy, rather than any constitutional...*
- *Emotional response by whom?*
- *I guess by persons considering the issue outside the legal context. The Constitution, as I see it, gives protection to people after birth.*

b. *Before addressing this claim, we feel it desirable to briefly survey, in several aspects, the history of abortion, for such insight as that history may afford us, and then to examine the state purposes and interests behind the criminal abortion laws.*

which provide ground for utilitarian evaluations from the point of view of subjective opinion (in this case these are the synonyms for the observed above subjective evaluations based on visual perception *as I/we see it*).
Besides, such phrases as "troubling question, passionate argument, sensitive and emotional nature, vigorous opposite views, unsettled and unsettling issue, novel and shocking opinion, distress, distressful life and future" appeal to emotional and hedonistic evaluations based on inner feelings of humans:

a. *The Court's opinion brings to the decision of this troubling question both extensive and historical fact and a wealth of legal scholarship.*
b. *Since then, Americans have spent thousands of hours in passionate argument on this issue.*
c. *We forthwith acknowledge our awareness of the sensitive and emotional nature of the abortion controversy, of the vigorous opposite views, even among physicians, and of the deep and seemingly absolute convictions that the subject inspires.*
d. *Chief Justice Warren Burger has called a case that raises an unsettled and unsettling issue in American society, abortion.*
e. *[The Constitution] is made for people of fundamentally differing views, and the accident of our finding certain opinions natural and familiar or novel and even shocking ought not to conclude our judgment upon the question whether statutes embodying them conflict with the Constitution of the United States.*
f. *Maternity, or additional offspring, may force upon the woman a distressful life and future. Psychological harm may be imminent. ...There is also the distress, for all concerned, associated with the unwanted child, and there is the problem of bringing a child into a family, already unable, psychologically and otherwise, to care for it.*

Such evaluations are motivated by psychological experiences that do not depend on the man's will or self-control; N.D. Arutyunova [1] called these "the most individualized kind of evaluations". Such evaluations orient a person to accommodation in natural and social environments and to reaching comfort. In argumentation all this produces a subjective influence on the listener, launching his own mechanism of sense perception, and in its turn influencing his process of making decisions (axiological condition in decision-making).
The last analysed example with sense perception and hedonistic evaluations showed that such evaluations may house further semantic bases for sublimated and rational evaluations.
The idea of sublimated evaluations is rather new and is described in a number of works of Russian linguists. Sublimated evaluations (from Latin *sublimare* - elate, exalt, pinnacle) are also called humanised, for they are placed above evaluations based on sense-perception, "humanising" them. [5]. According to N.D. Arutyunova [1] "this type of evaluations is closely connected with the term **archetype**, i.e. norm, example, potential requirements set towards the object". Rational evaluations, on the other hand, "are connected with practical activities, practical interests and everyday experiences of people. Their basic criteria is physical and psychological good, oriented at reaching a certain goal, performing a certain function, meeting certain standards" [5].
Thus, the above mentioned evaluations due to their argumentative influence,

besides the immediate impact on the interlocutors through their direct sense-perception evaluative meanings, acquire additional meanings, which reveal in the topics under consideration new ethical and humanistic values.
"We face re-orientation of sense-perception, psychological and rational evaluations towards spiritual values, so that in their turn these evaluations serve as semantic bases for sublimated evaluations" [5]. Trying to prove to the opponents the truth of their positions, speakers resort to these complicated evaluative constructions appealing to the listeners' normative and ethical ideals.
Besides, among major conditions for choosing evaluative lexical units in arguments, there exist other factors such as social role-status of interlocutors, their emotional state at the moment of speaking, and context.
In the court room role statuses are very well marked and the possible choice of speech acts is narrowed to restrained and rather high-flown style; evaluations in their turn are restricted to barely emotional. Therefore in the text we did not find explicitly expressed emotional, ethical, aesthetical, teleological, normative and utilitarian evaluations. Such types of evaluations are found only as based on other (sense perception and hedonistic) evaluations. Emotional state of interlocutors and verbal context are defined by a wider, extralinguistic context, i.e. corresponding to the requirements of a court session. Emotiveness finds vent through the use of moduses of visual, tentative and inner-sense perception. The context circumscribes the use of the language by resorting to relevant information only, without unnecessary emotiveness and unrelated sophistic casistry.

References

1. Arutyunova N.D. ed. *Logichesky analiz yazyka: yazyki dinamicheskogo mira* (*Logical Analysis of the Language: Language of the Dynamic World*), Dubna, 1999.
2. Baranov A.N., Sergeev V.M. Estestvenno-yazykovaya argumentatsiya v logike prakticheskogo rassuzhdeniya (Natural-Language Argumentation in the Logics of Practical Reasoning). In: *Myshlenie, kognitivnye nauki, intellekt* (*Thinking, Cognitive Sciences and Artificial Intelligence*), Moskva, 1988. pp. 104-119.
3. Brutyan G.A., *Argumentatsiya (Argumentation)*. Yerevan, 1984.
4. Gutton, Stephanie and Peter Irons, ed. *May It Please the Court: Arguments on Abortion*. The New Press, NY, 1995.
5. Pisanova T.V. *Natsional'no-kul'turnye aspekty otsenochnoi semantiki: eticheskie i esteticheskie otsenki* (*National and Cultural Aspects of Evaluative Semantics: Aesthetical and Ethical Evaluations*). - Moskva, 1997.
6. Richards I.A. & Ogden C.K. *The Meaning of Meaning: A Study of the Influence of Language upon Thought and the Science of Symbolism*. New York: Harcourt, Brace, 1930.
7. Stevenson Ch.L. *Facts and Values: Studies in Ethical Analysis*. - New Haven - London: Yale Univ. Press, 1964.
8. Tannen D. *The Argument Culture. Stopping American War of Words*. The Ballantine Publishing Group, NY, 1998.

Getting Good Value

Facts, Values, and Goals in Computational Linguistics*

Michael A. Gilbert

Department of Philosophy, York University, Toronto, Canada
Gilbert@YorkU.ca

Abstract. This discussion is intended to amplify the role of various intentional components in computer-human communication. In attempting to further the degree to which human subjects and computers can disagree and communicate within the context of that disagreement, i.e., "argue," the ability to identify and classify various locutions as facts, values, or goals is crucial. I suggest that inquiry into the conceptual framework, context, the identification of pre-existing beliefs, values and goals, and the discussion of their priority will facilitate the sought after communication. Toward this end, I use the concept of 'field' and its identification to help distinguish the relevant categories.

The distinction between facts and values has bedeviled philosophers through the ages. Plato pointed out that facts are the things we never argue about, we just go and find out what they are. Aristotle distinguished between *logos, pathos* and *ethos* with the first having far more rigorous aspects than the latter two. His "practical" syllogisms have been every bit as fraught with contention and uncertainty as his purely deductive have been precise and unquestioned. And it was David Hume who emphasized that one cannot move from any number of facts to even the simplest value. What became forever known as the "Is-Ought Problem" stated that "is" statements, i.e., statements of fact, cannot lead, deductively, to statements of value unless a normative axiom or postulate is put in. Thus, for example, the myriad of facts about how kind grandmother has been to my son cannot, *in and of themselves*, lead to the conclusion that he ought telephone her more often. One inevitably needs the moral principle that, say, you ought be nice to those who are nice to you, to make the normative leap.

In the last century the most celebrated usage of the fact-value distinction was by the logical positivists. Logical Positivism, coming into its own as it did immediately after World War I, was in no small part a reaction to the metaphysical idealism, religious emphasis, and unreasoning nationalism of the time. The Logical Positivists felt that the saving of humanity required the abandonment of mysticism, religion, and sentiment as guiding principles and the adoption, in their place, of scientific principles and reasoning. [1]. The heart of their programme

* This work has been supported in part by Social Sciences and Humanities Research Council Grant Nr. 5 512790. I also want to express my debt to the referees who were generally very helpful to and tolerant of an outsider.

V.N. Alexandrov et al. (Eds.): ICCS 2001, LNCS 2073, pp. 989–998, 2001.
© Springer-Verlag Berlin Heidelberg 2001

was the adoption of the Principle of Verifiability [PV] which, in its essence was exquisitely simple: Those statements that had, *in principle*, a means of verification were factual and had truth values, while those that did not were "mere" poetry or expressions of emotion. As a result,

1. [6 + 7 = 13]

is a fact, while

2. "13 is a lucky number,"

comes down to

3. "I like 13."

Similarly,

4. "The CN tower in Toronto is 553.33m meters high,"

or

5. "The CN Tower is the world's tallest free standing structure,"

are facts and, therefore, true or false, while the statements,

6. "The CN Tower is an engineering marvel,"

or

7. "The CN Tower is a dreadful waste of money,"

are expressions of feeling and neither true nor false.

Logical Positivism met a very unusual fate for a philosophical theory: it simply died. The death blow came from Gödel's theorem, but other problems were already weakening the foundations. Nonetheless, the underlying premiss that there is an identifiable distinction that can be made between facts and values has had a major impact.

The real difficulty, however, goes beyond the issue of the ascendancy of science over poetry to the very nature of actual decidability itself. Certainly, (1) is true, but someone might object that, in fact, [6 + 7] really equals D because the hexadecimal system is essentially more basic than the metric. Also, we may agree that nothing is more basic than [1 + 1 = 2], but adding one drop of water to another drop of water yields one drop of water. Clearly, this is a sophism, but a telling one nonetheless.

Consider (4). As it is a classically empirical statement it is either true or false. We know how to determine the height of the Tower, so the statement expressing its height is a fact. But anyone really being careful about the statement will readily admit that a variety of factors will enter into the determination of the truth and falsity of (4). Factors such as the tools or methods used for measurement, the outside temperature, season and time of day might all result in different measurements. Can something as high and complex as the CN Tower

be measured once and for all, and, if so, then to what degree of accuracy? Well, for the guidebooks, it is not a problem, but for other purposes the assumption that the measurement can be accurate to .33 meters without a *ceteris paribus* is far fetched.

An alternate attack comes from precisely the opposite direction. While (7) may well be highly contentious and a question of values, what about (6)? Surely, one wants to say, "if anything is an engineering marvel, then the CN Tower is." Is the fact that we do not have scientific criteria for what constitutes an engineering marvel sufficient grounds for saying that (6) only has meaning as an expression of emotion? Not only that, but such statements have logical consequences and corollaries as well as identifiable presumptions that can be laid out just as clearly as measurement techniques.

The difficulty that brought down the PV was that the very distinctions on which it rested – empirical vs. non-empirical, testable vs. non-testable – were fraught with hidden theoretic assumptions without which tests and observations could not even begin. The philosophical approaches that succeeded Logical Positivism in the 1940s and '50s were language based and emphasized the role of language in both epistemology and metaphysics. Subsequently, in the 70s and 80s the impact of the philosophy of science, especially as characterized by Popper, Kuhn, Feyerabend [10,8,2] and others began to have an enormous influence on philosophical work. These views disdained, to one degree or another, the idea that facts are independently identifiable, independent, that is, of the theory in which they are identified. Rather, it was argued, the theory itself determines what is and is not acceptable (even existent) data, as well as determining what are the correct and legitimate modes of testing, confirmation, and measurement.

Obviously, if the identification of facts and values are dependent upon the theory being used, then a machine's ability to distinguish between them will run into difficulties. The Logical Positivist approach of identifying by means of testing falls apart if the very system being examined is what determines the means of testing. From the point of view of computational linguistics, it is not so much the issues raised in the philosophy of science that are telling, as the difficulties inherent in interactions where the category of statement is not well identified. But this, if embraced, may very well be the solution as opposed to the problem.

To say that the distinction between facts and values is theory dependent, and to accept that what is true, what is false, and how these are tested and separated is also determined by the very theory that might be under discussion might well point to a solution of sorts. Namely, the acceptance that there is no real theory independent difference between facts and values. That is, all truths, all facts and/or values, are relative to a theory, and, most importantly, *every user is a theory.*

If we think in terms of science and mathematics, then a theory is something quite precise and delimited. There are axioms, postulates, corollaries, consequences and a plethora of background assumptions. But a machine interacting with an individual is working with something much less precise. Certainly, an

individual person may hold any number of theories (not all of which may be consistent one with another,) but there will also be a multitude of non-theoretic and even non-reflective beliefs, values and goals that come along with the person. When dealing with persons the best approach is not to think in terms of theories, but of fields. Fields range from those that one inherits such as race, religion and native culture, to those that are learned and acquired such as occupation, individual interests and avocations. Every field brings with it assumptions regarding the world and what is valued in it. Very importantly, fields bring with them goals, priorities and values. Identifying an individual's field memberships is one way of identifying her/his goals and values. The term was first introduced by Toulmin [11] to account for those aspects of reasoning that varied from domain to domain, and those that did not. Eligible evidence, for example, varied according to fields as when, to cite an extreme example, biblical scripture citation is acceptable in one sort of argument but not in another. Modus Ponens and the Law of Non-Contradiction, on the other hand, are invariant across fields.

The concept of field may be expanded, however, beyond the categories of Data, Warrant and Backing that Toulmin was interested in. One can add these categories, notably values, goals, morals, dispositions, preferences, and so on, or, alternatively, see the additional material as contained in the backing. The most comprehensive discussion of this is by Willard [12] where the role of values is carefully laid out. Willard emphasizes that is not merely interests or some few goals and values that may be field dependent, but entire worldviews, complete ways of imagining how everything hangs together. This in the field of Cost Benefit Analysis, everything can be converted into a cash value, while for many people not involved with CBA, such equations are at best nonsensical and at worst anathema.

So, it should be clear that I am presenting a panacea for the difficulties associated with computer - user argumentation. To the contrary, fields are not simple entities, both as a result of their internal complexity the sheer range of material they cover, and for a number of other reasons as well.. First of all, it is very important to remember that any given person belongs to a wide number of fields, which are not always consistent one with another and can, indeed, conflict. A Parent, for example, wants high quality education readily available, while the same person is also a Taxpayer and does not want education costs too high. As if that were not enough to make the situation difficult, the obvious solution, identify the hierarchical ranking of a user's fields is not as simple as it sounds. Aside from that task simply being difficult because many users are not aware of their rankings, the hierarchy in question is liable to change depending on the context and setting. Who the user is speaking with, where the user is, and the role the user is playing in a given situation will impact on the hierarchy. And the icing on the cake is that even on direct questioning a user may not be aware of his or her priorities, a well known problem facing statistical surveyors.

When humans communicate with each other they typically share a number of fields. In fact, they are, more often than not, members of the same field and communicating within its parameters. The fields they do not share will generally not

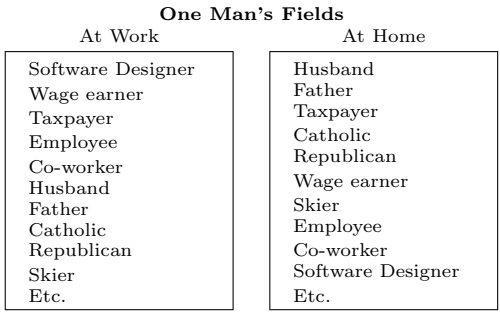


Fig. 1. The importance of an individual's fields will vary according to context and situation.

impinge, but may, of course, rear up at any moment. The point is that we most frequently argue with people we know and with whom we interact regularly. Only in relatively rare instances do we communicate in more than a superficial way with a complete and total stranger, and then when we do the initial part of the interaction focuses on finding common ground, i.e., shared fields. Frequently the very frame of the interaction, e.g., it's location or subject, can identify common fields as when one approaches an otherwise unknown salesperson. A computer, strictly speaking, has no fields and is forever interacting with people with whom it has nothing in common, and, most importantly, knows nothing about. But, given certain assumptions about why the User is interacting with the machine, and the objectives of the program, certain field commonalities may be assumed. That is, the program interface may be designed with any number of beginning assumptions about what the typical user will want, why they have begun the interaction, and what are the goals of both the User and the computer. Still, the identification of the subtleties of the User s field allegiance is a complex and intricate task.

Despite these difficulties and complexities the notion of fields and even the venerable Principle of Verifiability can be used to facilitate computer-user communications. I suggest several ways.

1. The machine can assume the existence of certain "shared" fields depending on the information context. (If the interaction concerns airplane bookings, then, e.g., the program will assume that the client is willing to board a plane, can afford a ticket, and so on.) As a result there may be tentative hypotheses made about common values.
2. Given the assumption of one or more fields, a loose version of the Principle of Verifiability can be assumed as a tool to separate facts and values using the computer's knowledgebase indexed by the field. (That is, the "facts" the computer relies upon given the presumed field sharing are taken to be shared unless interaction proves otherwise. The program assumes, e.g., that "Airplanes can fly" is a fact and not a value.)
3. The machine can inquire as to relevant field memberships not initially established and make highly friable assumptions on their basis. (The program might ask if the client has children, is employed, etc., and form one or more

queries based on values assumed to be shared by members of the identified fields.)

4. The computer can use field conflicts to determine the hierarchical ranking for an individual user. (A smoker who wants to travel but is unwilling to take a non-smoking mode of transportation is identifying a field hierarchical ranking.)

Rather than discuss these four proposals in isolation, I would like to use the time to discuss two examples in which they can come into play. The first one is a simple and familiar situation already quite common, while the second is more difficult and presents greater challenges.

In example one, which we will call the Air-Go[1] example, a user, Clara, is going to use the web based travel service, Air-Go, to book a reservation, a now familiar www experience. We will suppose that Clara lives in Boston, is attending a conference in San Francisco, wants to attach some holiday to the trip, and is on limited budget via her grant. This immediately identifies a set of fields to which Clara will *apparently* belong simply by virtue of arriving at the site and filling in the base information[2]. I call these fields the *Passive Context* as they, along with a plethora of potentially irrelevant data are passively assumed true unless otherwise contested.

Clara's Fields [Passive Context]: Air Traveler, East Coast Resident, Visitor to West coast

Obviously, this does not tell us a great deal. But Air-Go has a special feature called the Air-Go Travel Agent which engages the client in an inquiry aimed at determining the client's exact needs. In addition, and for the purposes of our example, most crucially, it inquires as to the purpose of the trip. It learns that Clara is going to a conference, but wants to spend some front end or back end time traveling. The program even asks which – front or back – is preferred, and learns that back end is preferred, but front end is acceptable. These fields now help to form the *Active Context* which can be shared more specifically between Clara and the program. Clara's fields are now extended to the following.

Clara's Fields [Active Context]: Air Traveler, East Coast Resident, Visitor to West Coast. Woman, Academic, Conference Attendee, Holidayer, Budget Conscious, Flexible Traveler.

Now the program can offer Clara a flight with dates she might not have considered, but which offer the lowest fares. In addition, Air-Go might try to tempt her with a particular hotel or spa that might meet her needs. Of course, nothing here is new. Companies frequently give questionnaires to their clients just to identify fields and interests and thereby tailor offerings to them. Or, companies track visits to web sites, check bookmarks, examine cookies, etc. to create an

[1] At time of writing there is no web site www.air-go.com, and this is used solely as an example.

[2] I say "apparently" because I do not want to confront the problems associated with misleading information, intentional or otherwise. The program can only, at this stage, assume honesty.

interest profile or stereotype of a user. In any case, the goal is to find out what fields an individual user belongs to so as to be able to make assumptions about their interests, which, four our purposes translates as beliefs, values and goals.

In the Air-Go example, there is no argumentative interaction. Clara may be offered a number of alternatives which she can accept or reject. No real attempt, at least in any site I ve visited, is made to pursue the matter. Imagine a hard sell program which inquired as to what was wrong with the offer, or which was more aggressive about tailoring to Clara s desires. I suggest we stop where we do because simply because we do not know enough about the User to be more persuasive. Perhaps this approach can be used to create a richer User Profile within the context of an interactive discursive exchange between a user and a program based on an item of contention. I.e., a computer can identify the beliefs, values and goals of a user through an interactive process and use that information to create a nuanced argumentative environment in which disagreement can be pursued. If we accept the essential idea of a field, and conclude that fields always contain within them facts, values, goals, and mores, then determining that a user is a member of the field permits the program to identify potential values. More, the program can even identify potential or actual field memberships which are likely to result in value conflicts which can then, in turn, be tested for their hierarchical status. Indeed, argument often proceeds effectively by bringing up value and goal conflicts. Being forced to choose between conflicting values indicates both a field hierarchy preference as well as something, a belief, goal or value, that must be abandoned or adjusted by the user.

A crucial difference between the commercial use of interest identification and the persuasive approach being explored is that in the argumentative context the goal of the process is to move the user from one belief or attitude to another. It is one thing to suppose that someone interested in air travel may also be interested in car rentals, another to conclude that a user interested in grade-school education will favour the release of public bond issues. In the former case the offer of rental cars is just ignored, in the latter the attempt to persuade may fall flat or even backfire.

In other attempts at dealing with computer-user argumentation [6] we have tackled a fairly thorny issue, viz., cigarette smoking or diet. In these examples an individual interacts with a computer programmed to try and change the user's attitude about a particular kind of behaviour. One problem we faced was determining the values an individual held and how those could be used to implement useful forms of argument that might lead to persuasion. I want to suggest now that the assumption that sets of values that form a nexus can be associated with membership in various fields, and, most importantly, these fields can be identified by the program through relatively simple interactions.

Imagine a program designed to persuade users to wear a seatbelt when in an automobile. A user seats himself down at a console and begins the interaction by choosing the subject of seatbelts. Now there are a multitude of arguments and thousands of reports and facts that can be brought to bear when building a case for the efficacy of seatbelts. But arguments rarely work if they are random

or take a shotgun approach. Rather, they are best tailored to the needs, beliefs, values and goals of the arguer. In other words, arguments are always geared to a particular audience [9]. The key is to identify the values held by the audience and utilized those to make the case for the values being propounded, in this case using seatbelts. It literally makes no sense to begin arguing with a User before knowing the user's position construed not as a simple discursive object such as a proposition, but rather as a nexus of goals, values and beliefs. To do otherwise is like attempting to write a coherent program by composing random bits of code.

Let us say that someone, call her Ursula, approaches a machine to discuss seatbelt wearing. Given that action we may make a number of simple assumptions about her[3]. First off, we can assume that Ursula does not generally wear seatbelts, but is at least occasionally in situations where their use would be expected. Further, we know that she is at least open enough both to the issue and as a person to engage in a persuasive interaction about the subject, *regardless of how ready she is to change her mind*. Beyond that, Ursula is a cipher. More specifically, we do not know which of many possible reasons are those that persuade Ursula to disdain seatbelt use.

When, in the natural course of events, we enter into an argument, or even a discussion, with someone, we make many assumptions about their beliefs and values. In addition, we make assumptions about the way the conversation is going to proceed [7]. Typically, if someone utters something like, "I don't like to wear seatbelts," or "I don't wear seatbelts," a good arguer will first try to find out why. As a rule, one never argues with a claim, but only with the reasons for a claim [4] to do otherwise is to lay yourself open for traps. Someone, for example, might want to ban hanging, a proposition with which you agree, but because it is too quick, not exactly your position. So, it is common in real argumentation for the first question to be, "Why?" How the machine parses this response, I leave to more schooled minds, but suffice it to say that the information is best coming from the User rather than being laid out within a range of choices.

Jut as in the commercial Air-Go example, the machine makes assumptions, but it can also learn something about the user, then the possible reasons for disagreement may be narrowed down. If the gender, occupation, hobbies, and social status of a user are known, then the possible arguments being presented can often be prioritized. Stereotypically, a male motorcycle fan will not offer breast discomfort as a reason, and a mother of four small children will not likely cite the thrill of danger[4]. So a little survey profiling the user can go a long way toward eliminating any number of reasons for disagreement.

Once an initial profile is created, the program can begin to inquire what aspect of the profile is the most crucial to the position User holds. In our example, the program quickly determines that User is a woman, mother, wife, part-time

[3] I am, for the purposes of this discussion, assuming that the user is not merely curious, obstreperous, or what have you. These are certainly possible, but they do not directly address the intended purpose of the machine.

[4] Naturally, anything is possible. But determining the values inherent in fields and audiences is always liable to correction.

teacher, not religious, and a theatre enthusiast who does not smoke. By using field associations that is, by looking for connections among the values pertinent to an identified field, the program decides what arguments might be persuasive. Being a wife and being a teacher are possibilities, but not as likely as being a woman or a mother. Someone who is religious might be amenable to an *argumentum ad verecundiam*, an argument from authority, and someone who is a non-smoker may be indicating a degree of risk-aversion that is relevant to the issue at hand. The computer may have associations between woman-hood and motherhood and seatbelt wearing that are usable in this context. As a result, it can now return the query, "Does your objection to seatbelt use have to do with a] being a woman, or b] being a mother, or c] neither," and, of course, this can be put into any language form desired.

When Ursula returns the information that it is her involvement in motherhood that moves her to avoid seatbelt use, the computer can go into that field and search for associations between motherhood and seatbelts. From research on the topic the program knows there are strong value rankings associated with children, helping children, and most relevantly to this issue, *saving* children. Continuing the inquiry, it soon becomes clear that User is afraid that if she is belted in she would be unable to assist her children in the event of a collision. Ursula's position has now been identified. *Effective argumentation, whether person to person or machine to person must begin with the correct identification of positions.* Failure to determine the position held by a user easily results in arguments that are beside the point, and may lead to making assumptions about values and beliefs that are not field specific.

Once having determined that the field Mother is the most relevant and highly ranked field for Ursula, it becomes possible to work from within those field values. That is, the program can assume the values of the field being used by the client, and determine if there are avenues of persuasion that rely on those values. Still keeping things simple, the computer might offer an argument designed to persuade Ursula that she has a better chance of rescuing her children from a collision if she had been wearing a seat belt herself. In other words, the computer accepts the value hierarchy of the field, and argues that the most valued result – keeping children safe – follows from the program's desired end and not the user's. This could be supported by facts concerning survival rates, and even information regarding her use of child safety seats. If Ursula gave up smoking so as not to die and leave her children motherless, then maybe she would begin to use her seatbelt for the same reason.

There is an analogous correspondence with "facts." Facts, for the purposes of a field, are those statements commonly believed by those who subscribe to the field. Thus, facts for astrologists are quite different than facts for astronomers. Mothers, for example, will place a strong reliance on their ability to "know," i.e., intuit, that their baby is ill. Within the Mother field the intuitional liaison between mother and baby is considered very strong, and data derived from it is taken very seriously. Knowing that this mode of verification plays an important role in the field, means the program can both be prepared to receive and to react

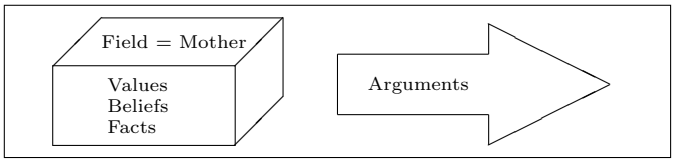


Fig. 2. When the field is identified, arguments may be drawn from within it.

to such arguments. (Vide, [4,5].) In a sense, the principle of verification is not being abandoned, but is rather being tailored to meet the needs and values of the field.

I am suggesting here that the identification of the fields to which an individual subscribes or is committed can provide an important method for identifying the values and beliefs of an individual user. In doing that two things become possible. First, the program is able to understand the position being put forward and place it in a frame with limited relevance associations, and secondly, arguments can be geared to the values and beliefs most important to the user. This means that by paying attention to fields a computer attempting to argue with a user can play a role similar to that played by another person, that is, it can make assumptions and put forward positions based upon a shared frame of reference. In addition, by identifying the field[s] of a user, the system can sort and filter informational databases so that "facts" used and sources referred to are those that one has a reasonable expectation of being acceptable to the user.

References

1. A.J. Ayer. *Language, Truth and Logic.* London : V. Gollancz, 1946.
2. P.K. Feyerabend. *Against method.* 3rd ed. New York : Verso, 1993.
3. M.A. Gilbert. Multi-Modal Argumentation. In *Philosophy of the Social Sciences.*, June 24: 2: 159-177.
4. M.A. Gilbert. *How to Win An Argument.* 2nd ed. New York: John Wiley & Sons, 1996.
5. M.A. Gilbert. *Coalescent Argumentation.* New Jersey: Lawrence Erlbaum Associates, 1997.
6. M.A. Gilbert, F. Grasso, L. Groarke, C. Gurr and J.M. Gerlofs. The Persuasion Machine: An Exercise in Argumentation and Computational Linguistics. In C.A. Reed, T.J. Norman and D. Gabbay, editors. *Argument and Computation.* (Forthcoming).
7. H.P. Grice. Logic & Conversation. In *Studies In the Way of Words.* Harvard U.P., Cambridge, MA, 1989. (orig., 1975).
8. S.T. Kuhn. *The structure of scientific revolutions.* 3rd ed. Chicago, IL : University of Chicago Press, 1996.
9. C. Perelman and L. Olbrechts-Tyteca. *The New Rhetoric: a treatise on argumentation.* University of Notre Dame Press, Notre Dame, Indiana, 1969. (orig Fr. 1958).
10. K. Popper. *Objective Knowledge: An Evolutionary Approach.* revised ed. Oxford: Oxford University Press, 1972/1979.
11. S. Toulmin. *The Uses Of Argument.* Cambridge: Cambridge UP, 1969.[Orig. 1958.]
12. C.A. Willard. *Argumentation & the Social Grounds of Knowledge.* Tuscaloosa: University of Alabama Press, 1983.

Computational Models of Natural Language Argument

Chris Reed[1] and Floriana Grasso[2]

[1] Department of Applied Computing, University of Dundee, Scotland
chris@computing.dundee.ac.uk
[2] Department of Computer Science, University of Liverpool, England
floriana@csc.liv.ac.uk

Abstract. This paper offers an introduction to the 2001 Workshop on Computational Models of Natural Language Argument (CMNLA 2001), a special event of the International Conference on Computational Science. The contributors to the workshop represent, in their backgrounds, the diversity of fields upon which the focus of the event draws. As a result, this paper aims not only to introduce the accepted papers, but also to provide a background that will be accessible to researchers in the various fields, and to sit each work into a coherent context.

1 Introduction

There is rapidly growing interest in the applications of argumentation for computer reasoning. One of the most mature and substantial research projects in the area is Pollock's OSCAR system [36], but the use of argumentation and dialectics in computational contexts has become increasingly popular, and with a variety of aims. Emphasis has been put on many aspects of the argumentation process: from the study on the structure of "valid" arguments [13,26], to the way complex arguments can be unreeled [9,48]. Argumentation is starting to play a key role in communication in multi-agent systems with significant results reported, inter alia, in [32]. Perhaps not by coincidence, argumentation theory itself has been enjoying a renaissance in recent years, demonstrated by the wide and extensive bibliography offered by [43].

Surprisingly, the ways in which arguments are rendered in natural language, the most obvious vehicle for the purpose, have not been investigated at depth, apart from few exceptions [10,28,31,37]. The endeavour is particularly challenging, both for computational linguists and for computer scientists, as argumentation is typically rich with rhetorical devices interacting at many different layers of abstraction, and is heavily dependent upon extra-linguistic context if it is to be successful. Moreover, a vast literature on both argumentation theory and rhetoric can offer great potential for exploitation, and, as in many other computational modelling problems, cross-disciplinary fertilisation is crucial for achieving significant results.

The 2001 Workshop on Computational Models of Natural Language Argument (CMNLA 2001) aims exactly at bringing this issue into the limelight.

V.N. Alexandrov et al. (Eds.): ICCS 2001, LNCS 2073, pp. 999–1008, 2001.
© Springer-Verlag Berlin Heidelberg 2001

The emphasis of the workshop is therefore squarely on computational models of natural language argumentation. As a result, the introduction and discussion presented here avoids the rich literature and fertile areas of research in which formal models of argumentation are employed for artificial reasoning, and computer-computer communication. Similarly, topics within the venerable field of argumentation theory in its own right do not here form a key focus of study, although it is inevitable that discussion at the workshop itself should draw on these fields. The discussion will instead focus on framing the problem, by providing a background that will be accessible to researchers in the various fields. The paper also aims to introduce the accepted papers, and, by emphasising the diversity of fields upon which the focus of the event draws, to sit each work into a coherent context.

1.1 Computational Models of Discourse

Computational Linguistics scientists concentrate on the "study of computer systems for understanding and generating natural language" [20], in order to develop "a computational theory of language, using the notions of algorithms and data structures from Computer Science" [1]. Typical problems for these research fields are [1]: How is the structure of sentences identified? How can knowledge and reasoning be modelled? How can language be used to accomplish specific tasks? Research spans, naturally enough, across the two ends of the human communication problem: understanding natural language, and producing it.

Understanding a piece of text is not merely a problem of "parsing" the sentences it comprises to identify the grammatical structure that can describe it. It is not even, or not only, to extract the "meaning" of such sentences, taking care of inherent natural language ambiguities. It is recognised that discourse has a much more complex structure than the simple collection of its sentences, and higher level factors must come into play in deciding whether a text is coherent. In other words, understanding text means primarily establishing whether such text had a purpose, in the mind of the writer, and how the purpose has been achieved by the writer, with the choice of one particular articulation of the sentences. Enlightening examples of such "rhetorical" parsing are in [29]. To achieve this level of comprehension, works on discourse annotation become of paramount importance, as they allow to gain a deeper understanding of how this presumed "structure" of a text is build up by human writers (a remarkable example for argumentative texts is [41]).

The same assumption that text has a structure, that can be determined, at least partially, by the writer's goal, has driven research in automatic natural language generation, for the complementary task of producing a piece of text in a human language from computer accessible, or, in general, non-linguistic representation of information [40]. If the text has to be produced for a purpose, and not by just gluing together sentences, knowledge is required of, at least, the domain of discourse, the purpose of the piece of text, the strategy to achieve such purpose, the rules for discourse organization, and, in order for the text to be tailored to a particular audience, the characteristics of the addressee.

Models of the structure of discourse are based on diverse hypotheses. For example, many of the early natural language processing systems have associated discourse with the concept of "recipe", or schema: if most of the texts containing an explanation (e.g. in an encyclopedia) have the same structure (start with the identification, "X is a Y", then pass to listing peculiar attributes of X etc.) then we can ascribe the same structure to all texts having the same purpose (to explain) [33]. Other systems are based on a different, less rigid approach: following Austin's intuition that utterances are performatives just as physical actions are [3], the metaphor of a "plan" has been used to describe discourse [2,22]. Just as a robot may have the goal of building a brick tower, and can decompose this problem into smaller and smaller tasks, until easily executable basic actions can be performed (lift arm, pick up block etc.), similarly a natural language tool may decompose a communicative goal into its steps, and use a plan based mechanism to achieve them. The original problem is then transformed into the problem of deciding how such communicative goals can be defined in the first place, and how they can be achieved, that is decomposed into smaller, more manageable problems. Guidance on these issues comes typically from discourse organisation theories, the most popular of which is perhaps the Rhetorical Structure Theory (RST) [27], a theory which, despite some criticisms [34], has been very widely used (see for example [24]).

1.2 Computational Models of Argumentative Discourse

The emerging area across computational linguistic and argumentation theory sets itself the task of investigating the forms and varieties in which arguments are put forward in natural language. Important issues are, therefore, those concerning the linguistic characteristics of argumentative texts: discourse markers, sentence format, referring expressions, style, and in general how natural language arguments can be produced on one hand, and recognised on the other. But equally fundamental are considerations on models of the argumentative process itself, in terms of characteristics of the *audience* [35], ultimate judge of the quality of an argument: their beliefs and propensities, but also emotions, personalities, values, norms. Preliminary reflections on this subject are included in [17]. The CMNLA 2001 workshop aimed at establishing common grounds for research, by gathering together researchers from very diverse and complementary fields, from computer science, to linguistic, to philosophy. Together, the contributors worked at defining the gaps that need to be bridged, while at the same time proposing avenues to bridge them.

2 The Gap at the Human Computer Interface

Work at the human-computer interface is confounded by two related challenges. The first arises as a result of human limitations, and the need to restrict, abridge, and limit, whilst at once explaining, repeating, and emphasising information in such a way that it can be successfully assimilated by the human. Computational

models of interaction, and particularly linguistic models of interaction, must necessarily take account of the limitations of the human interlocutor if they are to succeed. This consideration becomes all the more important in more demanding communicative situations, where, for example, the aim is to effect attitude change through persuasion, or to offer explanation through the expression of complex reasoning.

The second challenge is posed by the limitations of the communicating computer system, or, more accurately perhaps, by the traditional design of such systems. A sound and productive engineering approach has led to the development of natural language processing systems which build incrementally, one on the next. This much is to be valued most highly, and is taken as an indicator of a growing maturity in the field. It has also inevitably led to the widespread adoption of simplifying assumptions and a specific circumscription of the problem which are only now starting to be questioned. The underlying representation of linguistic and extra-linguistic knowledge, and the tasks which should properly be considered as part of the domain of Natural Language Processing are thus becoming important items on the research agenda in their own right.

Together, these two challenges represent a gap at the human-computer interface, comprising both the oversight of human limitations, and the underestimation of factors that need to be accounted for. Computational implementations of argumentation are one means of closing this gap. Theories of argument have squarely addressed complex communicative encounters involving persuasion and explanation, and offer insight into ways of circumventing, handling, and even exploiting, human limitations apposite in such situations. Furthermore, argumentation theory has, necessarily, embraced a much wider range of issues that impinge on communication. Similarly, computational systems focused specifically upon argumentation are necessitating a broader view of what constitutes Natural Language Processing, and are demanding reconsideration of some of the basic assumptions of the field.

2.1 Human Limitations

Following complex argumentation is an extremely demanding task for most people. If the single, simple deductive step embodied in the Wason selection task [45] can unerringly produce such dismal performance - and its manifold variations produce such wide discrepancies - then it should come as no surprise that comprehending substantial chains of argumentative steps represents a superbly intensive cognitive task. Even in cases in which the structure of argumentation is quite straightforward, the job of assimilating the data presented is not a simple one. Fox et al.'s [12] work on computer systems which reason and then present explanations in the oncology domain has clearly demonstrated that whilst formal models of argumentation provide a powerful mechanism for reasoning under conditions of uncertainty, the subsequent presentation of that reasoning to a human is highly troublesome.

In the first place, there are two conflicting forces which introduce a tension into the process of generating text involving arguments. On the one hand,

humans tire easily, becoming bored if information is repeated, or even if inferences are laboriously drawn out explicitly. This is reason for the ubiquity of enthymemes, in which components of an argument step are left implicit. Instead of carefully enumerating conclusion and minor and major premises of a Modus Ponens (as in The litmus paper turned red, and if litmus paper turns red then the solution is acid, so the solution is acid), one might more naturally offer an enthymematic version (The litmus paper turned red, so the solution is acid).

On the other hand, humans also have a tendency to get lost in an argument, and typically require a battery of linguistic devices to mark the way. A prime example is offered in the use of repetition and confirmation in informationally redundant utterances or IRUs, described by Walker [44]. IRUs have a range of functions including aid memory, support saliency, etc. but what concerns us here is the very idea that it is necessary at all to introduce components which convey no new information. Reed's [37] analysis of similar phenomena takes a related, if symmetrically opposite, approach whereby such utterances are not added to a text, but are instead always present at a deep level, and pruned as appropriate at the surface level - this analysis then supports explanation of differences in occurrences of enthymemes. For although enthymematic contraction is almost canonical in arguments structured around Modus Ponens, it is extremely rare in Modus Tollens constructions.

This tension between the need for brevity on the one hand and the requirements of waymarking on the other is one of the key issues addressed in Fiedler and Horacek [11]. The domain for their P. Rex system is the presentation of logical proofs. It is quite clear from their analysis that even in the relatively stylised linguistic usage of mathematical proof, it is crucial to be able to determine when an enthymeme (or other generic form of argument contraction) is appropriate, and when it is not. To render a proof comprehensible, information must be omitted where obvious or easily inferred (or where it occurs in particular configurations of particular syllogisms), but explained in detail where necessary.

The waymarking which is so vital to argument comprehension is perhaps best exemplified in the use of discourse markers (also, cues or clues). The use of specific lexical items marking argument structure was first discussed in the computational linguistics literature by Cohen [7], and has since played an important role in several models of natural language argument generation including [10,18, 38]. Carenini's GEA system [5], however, is one of very few which follows the task of generating the text of an argument from high level intentional goal through to textual output. Here, after selecting argument content, various components contribute to the ultimate lexical realisation, including processing to introduce appropriate discourse markers, aimed at making the reader's task that much easier - and the aims of the argument consequently that much more likely to be fulfilled.

In addition to these opposing forces working on the level of detail in an argument, and the introduction of discourse markers, there are also further considerations of human limitations and susceptibilities available to computer systems engaging argument. One is the technique of 'loading' or 'affect' - using biased or

evaluative terms to more or less subtly influence the reception of the argument. Hovy [23] included a study of affect in his PAULINE system; by altering values for a small range of stylistic variables (formality, force, etc.) the structure and lexical choice effected by the system could be radically altered. Gavenko's discussion [15] offers a broad analysis of particular forms of such affect including the use of visual and haptic metaphor. A computational model armed with the ability to flexibly employ such metaphors would significantly enhance the argumentation that could be generated by aiding the interlocutor in their task of comprehension.

2.2 Computer Limitations

To many in computational linguistics, and particularly in projects concentrating on argumentation, sentences like "The CN Tower is the world's tallest free standing structure", or, "The CN Tower is an engineering marvel", are typically represented in some knowledge base as p, or, q. Both are simply examples of 'propositions'. Gilbert [16] performs two tasks. The first is to remind us of the fact that such an abstraction is woefully inappropriate. Where the first sentence is (in the right context) a fact, the other (in the right context) is a value; the former does not admit of argumentation, whilst the latter does. Gilbert then discusses how the distinction between, and arrangement of, such facts and values is dependent upon the 'field' of context. Each user then represents not just one field, but a prioritized hierarchical arrangement of many different fields. This characterisation is similar to the trees of preferences used in systems such as [18] and [5], but Gilbert's suggestion goes further by associating with a given user not just a network of preferences (and values, and beliefs), but rather, a whole host of such networks, one for each role that the user might adopt. These networks are then themselves arranged in a hierarchy depending upon the context. Issues of inheritance and conflict between fields are going to play an important role in any system adopting this view of users, and these interactions are as yet ill-understood. There are, however, promising directions such as the application of context logic for capturing the closely related notion, due to Perelman and Ohlbrechts-Tyteca [35], of 'audience'. Preliminary work on this idea is to be reported in [8].

Perhaps the strongest tradition in widening the remit of natural language processing systems is in the incorporation of various non-textual information. One of the key pieces of work in the area is Maybury's system of 'communicative acts' which he applied both to the generation of argument [31] and to the inclusion of multimedia sources [30]. Green's analysis [19], examines how multimedia resources are employed in real argumentation, in an attempt to enrich automated generation of multimedia-rich argument. The structure of argumentation in non-textual source is notoriously difficult to identify in anything but the most trivial of examples, but recent work has made advances in this direction. Groarke's study [21], in particular, shows how art and editorial cartooning can be analysed to reveal deductive structure, and how that structure can then be assessed.

A third traditional restriction in the design of natural language processing systems is founded on intuitions about ethics and the role of the computer. Machines have not traditionally been equipped with the ability to deceive. And yet, it is becoming clear that in particular scenarios, such a capability is crucial. In negotiating (with one user, on behalf of another), or dealing with emotive issues (the oncological domain of Fox et al. [12] mentioned above is a rich source) there is a demand in the former case, for deception, and in the latter, for at least some equivocation. Recognition of the importance of the role of insincerity is what has motivated the work of Carofiglio and de Rosis [6], who build a probabilistic account based on Toulmin [42] structures of argument. Of course, as Carofiglio and de Rosis point out, deception carries with it a risk of discovery, which could be damaging in any communicative encounter. An analysis of cases of human deception in [4] demonstrates that equivocation plays a key role in salving the conscience and offering means of recovery in the event of discovery. The subtleties of linguistic prevarication employed in deception are beyond most current computational systems which typically exploit knowledge of a user's misconceptions (such as the 'licentious' flag in Zukerman et al.'s [47] NAG system). The analysis offered by Carofiglio and de Rosis provides a means to start refining the mechanisms which can be employed in the various forms of insincere communication.

Finally, a crucial challenge to be overcome if computers are to engage in dialogic argument is the problem of comprehension. Natural language understanding itself is an immense challenge of course, but understanding argument poses further problems above and beyond understanding the individual discourse elements. In order to understand and, ultimately, be in a position to be able to respond to human argument, it is essential that the large scale structure which provides the scaffolding is recognised accurately. This, after all, is one of the primary aims of the (primarily North American) undergraduate courses on argument analysis supported by texts such as [46]. The prospect of being able to automatically produce the sorts of diagrams available in, for example, Freeman's [14] analyses is extremely attractive. But to achieve that sort of competence, there are substantial problems upon which work has barely begun: automatic argument reconstruction, fallacy detection, the recognition of argument forms, and so on.

3 Prospects

The overlap between argumentation theory and Artificial Intelligence (AI) is particularly lucky as it can draw upon talented researchers in an extremely wide array of fields: linguistics, psychology, mathematics, cognitive science, philosophy, rhetoric, speech communication, politics, computer science, law, social science, economics, and more. The substantial work that can be achieved when members of these fields converge on argumentation was demonstrated dramatically at the Symposium on Argument and Computation, held in Scottish Highlands in June 2000. During the course of a week, two dozen scholars, most of whom

had not met previously, worked together to write a book examining the areas of interdisciplinary overlap. The remarkable results, which are to be published as [39], testify not only to the dedication of the contributors, but also to the level of interest that is being sustained in the area.

With a foundation like this, the prospects for argumentation and AI are extremely good - but there is a great deal of work to be done. Events like the Symposium on Argument and Computation and CMNLA are not only indicators of the interest within both fields in the crossdisciplinary overlap, but are also catalysts in stimulating new collaborations. The fruits of these collaborations hold very great promise.

Acknowledgements

We are most grateful to our colleagues who acted as reviewing committee, thus providing immense help in achieving a stimulating final programme:

- Cristiano Castelfranchi, Department of Communication Science, University of Siena, Italy.
- Fiorella de Rosis, Department of Informatics, University of Bari, Italy.
- Leo Groarke, Department of Philosophy, Wilfrid Laurier University, Waterloo, Ontario, Canada.
- Ehud Reiter, Department of Computer Science, University of Aberdeen, Scotland.
- Antoinette Renouf, Department of English Language and Literature, University of Liverpool, England.

We are also grateful to SIGGEN, the Association for Computational Linguistics Special Interest Group in Natural Language Generation, for kindly offering support-in-name to the workshop.

References

[1] J.F. Allen. *Natural Language Understanding*. The Benjamin/Cummings Publishing Company, Inc., 2nd edition, 1995.
[2] D.E. Appelt. *Planning English Sentences*. Studies in Natural Language Processing. Cambridge University Press, 1985.
[3] J. Austin. *How To Do Things With Words*. Oxford University Press, 1975.
[4] J.B. Bavelas, A. Black, N. Chovil, and J. Mullet. Truth, lies and equivocation. *Journal of Language and Social Psychology*, 9(1-2):135–161, 1990.
[5] G. Carenini. GEA: a Complete, Modular System for Generating Evaluative Arguments. In *(current volume)*.
[6] V. Carofiglio and F. de Rosis. Exploiting Uncertainty and Incomplete Knowledge in Deceptive Argumentation. In *(current volume)*.
[7] R. Cohen. Analyzing the Structure of Argumentative Discourse. *Computational Linguistics*, 13(1-2):11–24, 1987.
[8] J. Crosswhite, J. Fox, C.A. Reed, T. Scaltsas, and S. Stumpf. *Computational Models of Rhetoric*. In Reed et al. [39], (to appear).

[9] H. Dalianis and P. Johannesson. Explaining Conceptual Models - Using Toulmin's argumentation model and RST. In *Proceedings of The Third International workshop on the Language Action Perspective on Communication Modelling (LAP98)*, pages 131–140, 1998.
[10] M. Elhadad. Using Argumentation in Text Generation. *Journal of Pragmatics*, 24:189–220, 1995.
[11] A. Fiedler and H. Horacek. Argumentation in Explanations to Logical Problems. In *(current volume)*.
[12] J. Fox and S. Das. A unified framework for hypothetical and practical reasoning (2): lessons from medical applications. In D. Gabbay and H.J. Olbach, editors, *Practical Reasoning: Proceedings of FAPR'96*. Springer-Verlag, 1996.
[13] J. Fox, P. Krause, and M. Elvang-Goransson. Argumentation as a General Framework for Uncertain Reasoning. In D. Heckerman and A. Mamdani, editors, *Proceedings of the 9th Conference on Uncertainty in Artificial Intelligence*, pages 428–434. Morgan Kaufmann Publishers, 1993.
[14] J.B. Freeman, editor. *Dialectics and the MacroStructure of Argument*. Foris, 1991.
[15] S. Gavenko. Analysis of the Argumentative Effect of Evaluative Semantics in Natural Language. In *(current volume)*.
[16] M.A. Gilbert. Getting Good Value: Facts, Values and Goals in Computational Linguistics. In *(current volume)*.
[17] M.A. Gilbert, F. Grasso, L. Groarke, C. Gurr, and J.M. Gerlofs. *The Persuasion Machine: An Exercise in Argumentation and Computational Linguistics*. In Reed et al. [39], (to appear).
[18] F. Grasso, A. Cawsey, and R. Jones. Dialectical Argumentation to Solve Conflicts in Advice Giving: a case study in the promotion of healthy nutrition. *International Journal of Human-Computer Studies*, 53(6):1077–1115, 2000.
[19] N. Green. An Empirical Study of Multimedia Argumentation. In *(current volume)*.
[20] R. Grishman. *Computational Linguistics : an Introduction*. Studies in Natural Language Processing. Cambridge University Press, 1986.
[21] L. Groarke. Logic, art and argument. *Informal Logic*, 18(2-3):105–129, 1996.
[22] B.J. Grosz and C.L Sidner. Plans for Discourse. In P. Cohen, J. Morgan, and M. Pollack, editors, *Intentions in Communication*, chapter 20, pages 417–444. MIT Press, Cambridge (Mass.), 1990.
[23] E. Hovy. Pragmatics and Natural Language Generation. *Artificial Intelligence*, 43:153–197, 1990.
[24] E. Hovy. Automated Discourse Generation using Discourse Structure Relations. *Artificial Intelligence*, 63(1-2):341–385, 1993.
[25] K. Jokinen, M. Maybury, M. Zock, and I. Zukerman, editors. *Proceedings of the ECAI-96 Workshop on: Gaps and Bridges: New directions in Planning and NLG*, 1996.
[26] N. Karacapilidis. An Argumentation Based Framework for Defeasible and Qualitative Reasoning. In Jokinen et al. [25], pages 37–42.
[27] W. Mann and S. Thompson. Rhetorical Structure Theory: Toward a Functional Theory of Text Organization. *Text*, 8(3):243–281, 1988.
[28] D. Marcu. The Conceptual and Linguistic Facets of Persuasive Arguments. In Jokinen et al. [25], pages 43–46.
[29] D. Marcu. *The Theory and Practice of Discourse Parsing and Summarization*. MIT Press, 2000.
[30] M. Maybury. Planning multimedia explanations using communicative acts. In *Proceedings of the 9th National Conference on Artificial Intelligence (AAAI91)*, pages 61–66. AAAI, MIT press, July 1991.

[31] M. Maybury. Communicative Acts for Generating Natural Language Arguments. In *Proceedings of the 11th National Conference on Artificial Intelligence (AAAI93)*, pages 357–364. AAAI, AAAI Press / The MIT Press, 1993.

[32] P. McBurney and S. Parsons. Agent ludens: Games for agent dialogues. In P. Gmytrasiewicz and S. Parsons, editors, *Proceedings of the Third Workshop on Game-Theoretic and Decision-Theoretic Agents (GTDT2001)*, AAAI Spring Symposium. AAAI Press, Menlo Park, CA, 2001.

[33] K. McKeown. *Text Generation: Using Discourse Strategy and Focus Constraints to Generate Natural Language Texts*. Studies in Natural Language Processing. Cambridge University Press, 1985.

[34] J. Moore and C. Paris. Planning Text for Advisory Dialogues: Capturing Intentional and Rhetorical Information. *Computational Linguistics*, 19(4):651–695, 1993.

[35] C. Perelman and L. Olbrechts-Tyteca. *The New Rhetoric: a treatise on argumentation*. University of Notre Dame Press, Notre Dame, Indiana, 1969.

[36] J. Pollock. OSCAR: a General Theory of Rationality. In R. Cummins and J. Pollock, editors, *Philosophy and AI: Essays at the Interface*, chapter 9, pages 189–213. MIT Press, Cambridge (Mass.), 1991.

[37] C.A. Reed. The Role of Saliency in Generating Natural Language Arguments. In T. Dean, editor, *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence (IJCAI'99)*, pages 876–881. Morgan Kaufmann Publishers, 1999.

[38] C.A. Reed and D.P. Long. Generating the structure of argument. In *Proceedings of the 17th International Conference on Computational Linguistics and 36th Annual Meeting of the Association for Computational Linguistics (COLING-ACL'98)*, pages 1091–1097, 1998.

[39] C.A. Reed, T.J. Norman, and D. Gabbay, editors. *Argument and Computation*. (to appear).

[40] E. Reiter and R. Dale. Building Applied Natural Language Generation Systems. *Natural Language Engineering*, 3(1):57–87, 1997.

[41] S. Teufel, J. Carletta, and M. Moens. An Annotation Scheme for Discourse-Level Argumentation in Research Articles. In *Proceedings of EACL*, 1999.

[42] S. Toulmin. *The Uses of Argument*. Cambridge University Press, 1958.

[43] F. van Eemeren, R. Grootensdorst, F. Henkemans, J.A. Blair, R. Johnson, E. Krabbe, C. Plantin, D. Walton, C. Willard, J. Woods, and D. Zarefsky. *Fundamentals of Argumentation Theory: A Handbook of Historical Backgrounds and Contemporary Developments*. Lawrence Erlbaum Associates, Hillsdale, NJ, 1996.

[44] M.A. Walker. The Effect of Resource Limits and Task Complexity on Collaborative Planning in Dialogue. *Artificial Intelligence*, 85(1-2):181–243, 1996.

[45] P. Wason. Reasoning. In B.M. Foss, editor, *New Horizons in Psychology*. Harmondsworth: Penguin., 1966.

[46] B.A. Wilson. *The Anatomy of Argument*. University Press of America, 1980.

[47] I. Zukerman, K. Korb, and R. McConachy. Perambulations on the Way to an Architecture for a Nice Argument Generator. In Jokinen et al. [25], pages 32–36.

[48] I. Zukerman, R. McConachy, and K. Korb. Using Argumentation Strategies in Automated Argument Generation. In M. Elhadad, editor, *Proceedings of the 1st International Conference on Natural Language Generation (INLG-2000)*, pages 55–62, 2000.

An Empirical Study of Multimedia Argumentation

Nancy Green
Department of Mathematical Sciences, University of North Carolina at Greensboro
Greensboro, North Carolina 27402, USA
nlgreen@uncg.edu

Abstract. We have analyzed a corpus of human-authored arguments expressed in text and information graphics, non-pictorial graphics such as bar graphs. The goal of our research is to enable intelligent argument generation systems to make effective use of these media. This paper presents and compares two classification schemes used to analyze the corpus, illustrated by examples from the corpus, and discusses implications for generation systems.

1 Introduction

In many domains of discourse, arguments employ quantitative data for support. Frequently, these arguments are expressed in a combination of text, tables of statistics, and information graphics such as bar graphs. The goal of our research is to enable intelligent argument generation systems to make effective use of text and information graphics. Towards this goal we are making an empirical study of the relationship of information graphics to text in a corpus of arguments. We have analyzed the corpus in terms of text coherence relations and argumentation strategies. In this paper, we survey some of the results of our work so far and discuss the implications for argument generation systems.

2 Related Work

In this section we describe related work in multimedia generation, focusing on issues relevant to computer generation of arguments in integrated text and information graphics.

Since the 1980's research in intelligent multimedia generation systems (IMPS) has addressed the problem of generating integrated text and graphics [1]. Early systems that generated text and pictorial graphics (illustrations, diagrams, and maps) identified several key issues in multimedia generation [2],[3],[4]. The media selection problem is the problem of selecting use of text or graphics for parts of the presentation. The simplest approach to media selection is for the choice to be built into the system, e.g., by encoding media-specific information into rhetorical strategies. A limitation of this approach is loss of flexibility and expressiveness. Another key issue, media coordination, is the problem of coordinating parts of a presentation expressed in different media. Early work in this area included generating multimodal referring expressions, references to things in the world made through a combination of natural language referring expressions and pictorial representations [2],[5]. These approaches required reasoning about the intended effects of components of the presentation in

V.N. Alexandrov et al. (Eds.): ICCS 2001, LNCS 2073, pp. 1009-1018, 2001.
© Springer-Verlag Berlin Heidelberg 2001

different media. A related issue is the problem of detecting and resolving unintended effects [6].

Other IMPS research addressed the problem of generating natural language captions for computer-generated information graphics. Information graphics are non-pictorial graphics such as scatter plots and bar charts [7]. PostGraphe partially automated the generation of business reports [8]: the user could select a template indicating the main point of his desired report, e.g., to compare profits of one company to another; Postgraphe then would search among a set of graphic designs to create a graphic emphasizing the user's selected point, and would generate a caption based upon the selected design and template. Another system, the Caption Generation System [9] automatically generated captions for information graphics produced by a powerful automatic graphic design system. The captions were intended to help the user to interpret a graphic by describing complex aspects of its design and relationships between the graphic's elements and the user's data.

In contrast, the text generated by AutoBrief played a central rather than a supporting role in achieving the goals of a presentation [10],[11]. AutoBrief generated advisory presentations on transportation scheduling in integrated text and information graphics. In the AutoBrief architecture, the first stage of processing was to generate a media-independent plan for achieving presentation goals, without specifying how the acts of the plan would be realized in a particular medium. In the next stage, media selection, parts of the plan were assigned to the system's text generator and/or automatic graphic design component for realization. The AutoBrief architecture enabled the information graphics to be designed to achieve complex communicative goals. However, although some types of presentations generated by AutoBrief could be analyzed in terms of argument strategies, presentation plan structure in AutoBrief did not explicitly represent this type of knowledge. The need to consider argument strategies in media selection and media coordination was argued in [12]. Also, a later implementation of the AutoBrief architecture in a system for generating evaluative arguments explicitly represented evaluative argument strategies; the representation was used to influence several aspects of text generation such as lexical choice [13]. The goal of our corpus analysis is to gain knowledge that can inform the design of future generation systems such as AutoBrief. Previous corpus studies of multimedia documents [14],[15] focused on different properties than those that we studied.

3 Corpus

The focus of our corpus analysis is on use of information graphics in arguments intended for an educated reader who is not necessarily an expert in the domain of discourse. The present corpus consists of excerpts from the following :

- A report prepared by the U. S. Department of Energy on the topic of global warming, published as a hypertext document on the world wide web [16],
- Articles on bird census data from several issues of a newsletter for non-scientists, prepared by a research institution and published as a print quarterly [17],
- A college textbook on software process improvement for undergraduate computer science students [18].

Excerpts were selected that expressed an argument partly in text and partly through use of one or more information graphics or maps. We then analyzed the excerpts in two

ways: first, in terms of coherence relations, and second, in terms of higher-level argumentation strategies. Since the two approaches led to different insights, we shall discuss them separately.

4 Analysis of Coherence Relations

The goal of this analysis was to investigate the types of relations between text and graphics in our corpus as indicated by cross-media cue usage. We define a *cross-media cue* as a phrase used to signal explicitly a relationship between some text and a graphic in the same document, e.g., "Figure 9.4 shows that", "(Figure 9.4)", "(See Figure 9.4)". Cross-media cues are similar in some respects to discourse cue phrases, connectives such as 'although' that help to convey discourse coherence relations, semantic and pragmatic relations between units in a coherent text. First, both help the reader to connect parts of a presentation, although in many cases the reader may be able to infer the connection. Second, for both types of cue, the same construction may be used to perform different functions, and the same function may be expressed in different ways. The expression of cross-media cues may involve use of discourse deixis, use of an expression that refers to the physical layout of a document, e.g., "the above Figure" [19]. Also, multimodal referring expressions can be used to indirectly signal a relationship between text and a graphic. However, we only considered explicit cues.

Rhetorical Structure Theory (RST) provides a set of coherence relations that can be used to describe the structure of a monomodal text [20]. According to RST, a common structural pattern is formed by two adjacent spans of text related by a coherence relation. One of the spans (the *nucleus*) is more essential to the writer's purpose than the other (the *satellite*). We found that several RST relations could be used to describe the relationship between the graphic referred to in a cross-media cue and the rest of the sentence, provided that the graphic was analyzed as the nucleus of the relation and the rest of the sentence as the satellite. In the next section, we give examples of the relations that we found in our corpus, and the definition of the relation (as we adapted it to describe the multimedia discourse).

4.1 Coherence Relations

Preparation. Text prepares the reader to expect and interpret the graphic. Example: "Figure 1 on page 3 shows how average numbers of four common species have changed over the course of Project FeederWatch."

Restatement. Text restates some or all of the situation expressed in the graphic. Example: "In the United States, nearly 85 percent of anthropogenic greenhouse gas emissions result from the burning of fossil fuels (Figure 3)." The figure contains a pie chart in which the slice denoting the contribution from the United States is labelled "85%".

Summary. Text summarizes multiple pieces of information expressed in the graphic. There were two varieties of summary in the corpus. In the first example, the text makes a generalization about a subset of the data shown in the graphic, and in the second

example, the text expresses an arithmetic summarization of the raw data shown. Examples: "From Fig. 9.5, you can see that the numbers of test defects are much lower in the later programs.", "Current projections show U.S. emissions increasing by 1.2 percent annually between 1995 and 2015 absent any policy interventions (see Figure 4)."

Evaluation. Text provides an evaluation of the situation conveyed by the graphic. Example: "In particular, anthropogenic carbon dioxide emissions have increased dramatically … (Figure 1)"

Elaboration. Text provides additional information about the data not given in the graphic. Example: "These data, shown in Figs. 9.1 and 9.2, are for the times to both find and fix these defects."

4.2 Discussion

This analysis shows a variety of relations between text and graphic that can be signaled by cross-media cue usage. However, it raises some issues for natural language generation. In general, what makes a multimedia document coherent? For example, are the relations between cross-media cue and graphic constrained by the coherence relations between spans of text in the document? Are there any relations needed to characterize multimedia discourse coherence that are not needed for monomodal text? We assume that these questions are relevant to argument generation, i.e., that in order to be effective, the presentation of an argument must be coherent. Another question is what factors determine where to place a cross-media cue? An overly simplistic approach might be to add a cross-media cue whenever a graphic stands in one of the above relations to the text. However, this approach is not sufficient when more than one unit in the text is related to the same graphic. In many cases, a graphic is relevant to more than one sentence. To address this problem, it is necessary to analyze the higher-level organization of the text, which is addressed in the next section.

5 Analysis of Argument Strategies

The goal of this analysis was to investigate the roles of text and graphics in argument strategies. We examined the corpus for instances of strategies described in a textbook on writing effective arguments [21]. The results can be summarized as follows:

1. A single graphic can be designed to convey more than one part of the same argument strategy, or to convey parts of several strategies.
2. The same part of an argument strategy may be expressed in both text and graphics.
3. In addition to bearing the argument proper, text in the body of a document may be what we shall call *commentary-bearing* rather than *argument-bearing*. The former category includes comment on
   - a graphic's role in the argument, by means of cross-media cues;
   - the location of the graphic in the document, by means of discourse deixis;
   - correspondences between graphical elements and database elements, e.g., the type of data shown in a graph; and

- salient visual features of the graphic, e.g., a sharp change in slope of a line in a line graph.

4. Placement of this commentary-bearing text may be interleaved with argument-bearing text. Furthermore, cross-media cues may occupy sites that would otherwise be available for discourse cue phrase placement.
5. Intuitively, use of a graphic can add to some dimensions of a presentation's effectiveness, e.g., comprehensibility, memorability, or persuasiveness. (There is some support for these intuitions from cognitive psychology also.)

5.1 Argumentation Strategies

We now present detailed examples from the corpus to provide evidence for these observations. Excerpts are given in tabular format. The excerpt is shown in the order in which it appears in the source. The third column contains notes from our analysis. In the third column, **->** and **//>** denote *supports* and *refutes*, respectively; C denotes *commentary* and G the *graphic* referred to in the text. Each table is accompanied by a figure with a schematic of the design of any relevant graphics in the source.

Addressing the Counterargument. The text shown in Table 1 makes use of this stragegy in units (5-7). Support is provided for a claim (5) by acknowledging (6) and refuting (7) a possible counterargument (not explicitly stated in the text; denoted by *ctr(5)* in the table). Both of the graphics in the source have the schematic form shown in Figure 1. They express the argument given in units (5-7); they show data supporting both (6) and (7), but (7) is emphasized over (6) by graphic design and layout. The design emphasizes Phase over Defect Type by encoding Phase by position on the x-axis for each cluster of bars and Defect Type by shading. The layout emphasizes Phase over Language by presenting data for each language in a separate graphic on different pages. Finally, note that a cross-media cue placement algorithm based only coherence relations between text and graphics would overgenerate; it would generate the same cue for each of (5), (6), and (7), since each provides a summary of data in the two graphics.

Table 1. Source: [18], p. 275

Unit	Excerpt	Analysis
1	**defect identification costs are highest during test and use.**	1 -> 2
2	**Thus anyone who seeks to reduce development cost or time should focus on preventing or removing defects before starting test.**	Main claim
3	**This conclusion is reinforced by the PSP data on the fix times for 664 C++ defects and 1377 Pascal defects …**	C
4	**These data, shown in Figs. 9.1 and 9.2, are for the times to both find and fix these defects.**	C
5	**Fix times are clearly much longer during test and use than in the earlier phases.**	5 -> 2, [6,7] -> 5
6	**While this pattern varies somewhat between the two languages and by defect type,**	6 -> ctr(5), G -> 6
7	**the principal factor determining defect fix time is the phase in which the defect was found.**	7 //> ctr(5), G -> 7

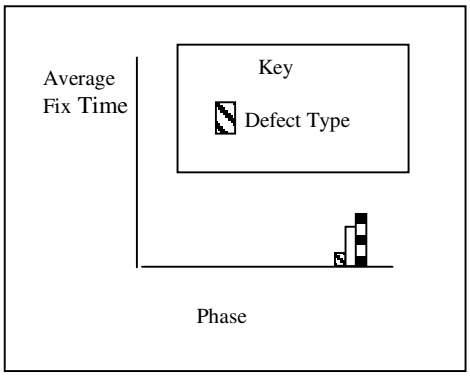


Fig. 1.

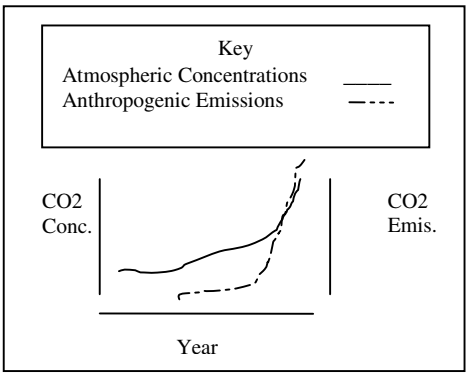


Fig. 2.

Arguing for a Causal Relation by Showing Correlation (first example). The text shown in Table 2 makes use of this strategy in units (1-6). The main claim, that something (4) caused a changing situation (1) during a certain time span (2), is supported by claiming that there was a proportional change in a related condition (5) over the same time span (6). The graphic has the schematic form shown in Figure 2. It shows data supporting both the claim in units (1-2) and units (5-6). In addition, the temporal correlation of (1) and (5) is shown by plotting the two sets of data against the same x-axis, which encodes time. The evaluation that anthropogenic emissions "increased dramatically" is reinforced by the steep rise in the line representing emissions.

Table 2. Source: [16]

	Excerpt	Analysis
1	**Atmospheric concentrations of several important greenhouse gases (…) have increased by about 25 percent**	G-> [1-2]
2	**since large-scale industrialization began some 150 years ago.**	Time span of 3
3	**The growth in their concentrations is believed to be caused by**	Main claim
4	**human (anthropogenic) activity.**	Main claim
5	**In particular, anthropogenic carbon dioxide emissions have increased dramatically**	[5-7] -> [3-4] G-> [5-6]
6	**since the beginning of the industrial age**	Time span of 5
7	**due largely to the burning of fossil fuels and forestation (Figure 1).**	Caused [5-6], C

Arguing for a Causal Relation by Showing Correlation (second example). The text shown in Table 3 makes use of this stragegy in units (1-5), although it uses a graphic in a different way than shown in the preceding example. The main claim is a yes-answer to the rhetorical question (1). It is supported by claiming that a trend described in (2)

has features described in (4) that are correlated with events described in (5). (In addition, the trend in (2) is contrasted with the trend in (3) to show that winter weather affects northern wren populations differently than southern wren populations.) The graphic has the schematic form shown in Figure 3. It shows the trends in (2) and (3). The temporal correlation of features (4) of the trend in (2) with events in (5) is shown by annotating the peaks in the line denoting N.E. population with arrows and labels denoting the events ("snowstorms").

Table 3. Source: [17], Autumn 1994, p. 5

	Excerpt	Analysis
1	**Did the bitter cold and frequent snowstorms this past winter take a similar toll on wrens in the Northeast?**	Main claim: yes
2	**We examined the weekly FeederWatch counts and noted that, in the Northeast Region, Carolina Wrens visited fewer and fewer feeders as the season progressed,**	G -> 2
3	**although they didn't follow this pattern in the Southeast region (Figure 7).**	G -> 3, C
4	**The Northeast decline came after a sharp peak in visitation during the week of January 8-14.**	G -> 4
5	**This period corresponded with a severe storm that dropped at least three feet of snow over most of the Northeast and glazed much of the Southeast with ice.**	Annotations in G

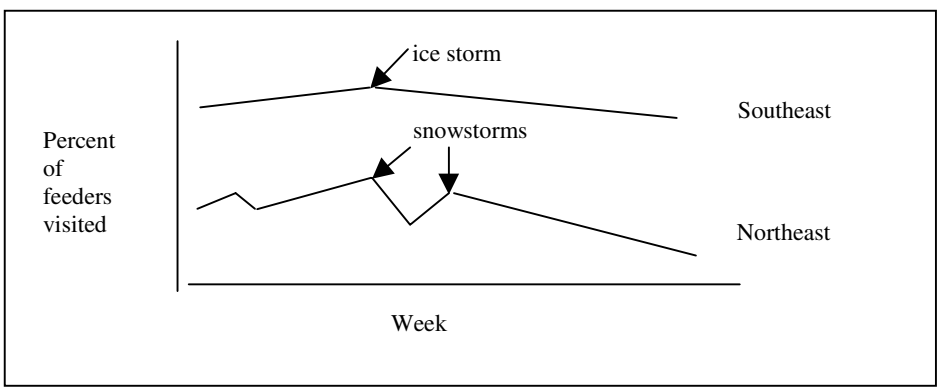


Fig. 3.

Inductive Generalization. The text shown in Table 4 makes use of this strategy more than once in units (1-5). A previous inductive generalization (1) was based on limited data described in (3). However, new data described in (4) falsify (1). Together, the old and new data support a new inductive generalization (2). Both graphics for the two whale species have the schematic form shown in Figure 4. The leftmost bar chart

shows data referred to in (3) supporting (1), while the rightmost bar chart shows data referred to in (4) falsifying (1); together the data in the two bar charts support (2). (5) repeats the argument in (3-4) for finback whales. In addition to the bar charts, the graphic contains a geographic representation of the parts of the Northern Hemisphere; arrows point to the region from where the data shown in the bar chart was collected.

Table 4. Source: [17], Summer 1995, p. 3

	Excerpt	Analysis
1	**During the first year that BRP studied the world's two largest whale species (blue and finback whales), our data suggested that they sang only at certain times.**	Old claim
2	**As it turns out, these species are vocally active all year round.**	[3-5] -> 2
3	**Initially, we monitored the region between Newfoundland and the Caribbean margin. There, we detected the most whale calls during the six-month-long winter breeding season.**	G -> 3
4	**More recently, however, we've recorded blue whales in the northeast Atlantic and we hear few or no calls in the winter there. Instead, the number of calls increases through late spring, peaks in late summer, and then decreases again in the fall – almost a mirror-image of the pattern farther to the south and west (Figure 1, top).**	G -> 4. C
5	**Finback sounds follow a similar pattern (Figure 1, bottom).**	G -> 5, C

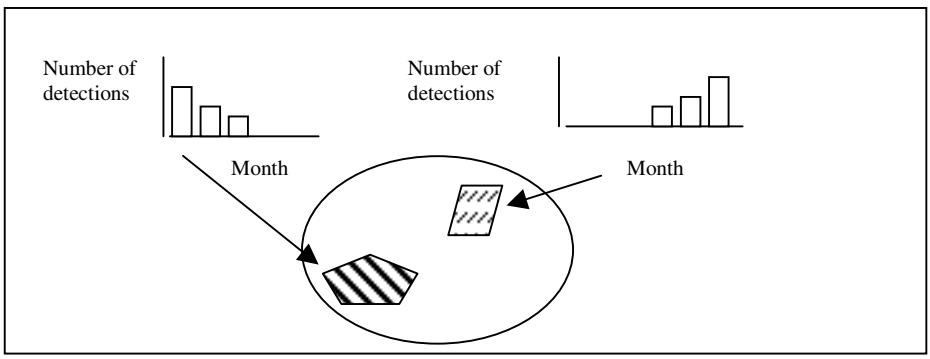


Fig. 4.

Falsification of Hypothesis of Sufficient Cause. The text shown in Table 5 makes use of this stragegy in units (2-3) to argue that a condition (2) is not a sufficient cause of a situation described in (3) by claiming that the condition was present but that the situation did not occur. The graphic has the schematic form shown in Figure 5; the

bottom half of the figure, labeled *Figure 3b* shows data supporting the claim that the situation did not occur. Although the top half of the figure, labeled *Figure 3a*, is used with (4) as part of another argument strategy (addressing a possible counterargument), by juxtaposing it with Figure 3b the author enables the reader to see a striking visual contrast in the shape of the lines in the two graphics. Intuitively, seeing the two together helps to persuade the audience that (3) is an accurate interpretation of the data shown in Figure 3b.

Table 5. Source: [17]

	Excerpt	Analysis
1	**… the "every other year" rule has its exceptions.**	[2-3] -> 1
2	**Even though conifers seems to produce abundant seeds only every second year,**	
3	**two conifer specialists, White-winged and Red crossbills, do not stage invasions in alternate years (see Figure 3b below).**	G -> 3, C
4	**Last winter was exceptional because both the regular biennial invaders (Figure 3a) and the less predictable crossbills traveled south together.**	G -> 4. C

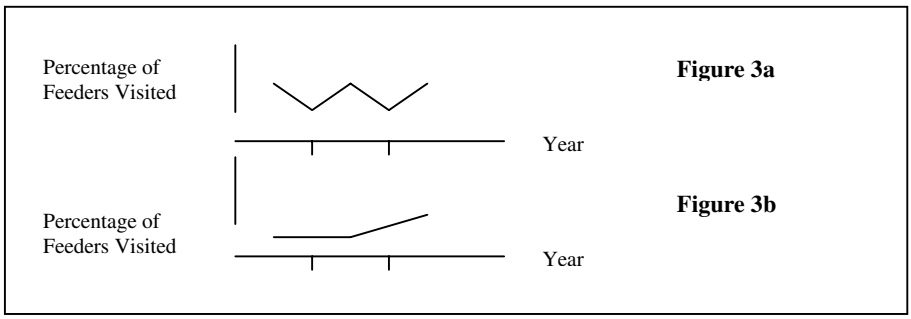


Fig. 5.

5.2 Discussion

This analysis has important implications for multimedia argument generation architectures. The influence of argument strategy can be seen in both text and graphics in the corpus. Thus, in a multimedia generation architecture such as described above, it would make sense to represent argument strategies in the media-independent plan explicitly. For example, this would enable media selection and graphic design processes to be informed by the underlying argument strategy. Second, we showed that the text serves two kinds of functions: to advance the argument proper and to help the reader to interpret a graphic and recognize its intended contribution to the argument. This suggests adding a post-graphic-design text-only planning phase to the architecture. This phase would be responsible for planning commentary-bearing acts (including cross-media cues) and integrating them with the already-planned argument-bearing acts previously allocated to text.

Acknowledgments

This research was sponsored by a UNCG Summer 2000 Excellence Research Award.

References

1. Roth, S.R., and Hefley, W.E. Intelligent Multimedia Presentation Systems: Research and Principles, in Maybury, M. T. (ed.). *Intelligent Multimedia Interfaces*. MIT Press, Boston, 1993, pp. 13-58.
2. McKeown, K.R., Feiner, S.K., Robin, J., Seligmann, D.D., and Tanenblatt, M. Generating Cross-References for Multimedia Explanation. *Proceedings of AAAI 1992*, 9-16.
3. Wahlster, W., André, E., Finkler, W., Profitlich, H.-J. P., and Rist, T. Plan-based integration of natural language and graphics generation. Artificial Intelligence 63(1993), 387-427.
4. Maybury, M. Planning multimedia explanations using communicative acts. *Proceedings of the Ninth National Conference on Artificial Intelligence,* 1991, 61-6.
5. André, E., and Rist, T. Referring to World Objects with Text and Pictures. *COLING-94*, 530-534.
6. Marks, J. and Reiter, E. Avoiding unwanted conversational implicatures in text and graphics. *Proceedings of the Eighth National Conference on Artificial Intelligence (AAAI-90)*, 450-6.
7. Card, S.K., Mackinlay, J., and Shneiderman, B. (eds.). *Readings in Information Visualization: Using Vision to Think.* Morgan-Kaufmann, 1999, ch. 1.
8. Fasciano, M., and Lapalme, G. Intentions in the coordinated generation of graphics and text from tabular data. *Knowledge and Information Systems*, Oct 1999.
9. Mittal, V., Moore, J., Carenini, G., and Roth, S. Describing Complex Charts in Natural Language: A Caption Generation System. *Computational. Linguistics, Special issue on Natural Language Generation.* Vol. 24, issue 3, (1998), 431-467.
10. Green, N., Carenini, G., and Moore, J. A Principled Representation of Attributive Descriptions for Integrated Text and Information Graphics Presentations. In *Proceedings of the Ninth International Workshop on Natural Language Generation*, 1998, 18-27.
11. Kerpedjiev, S., and Roth, S.F. Mapping Communicative Goals into Conceptual Tasks to Generate Graphics in Discourse. *Proceedings of Intelligent User Interfaces (IUI '00), New Orleans, LA, Jan 2000, pp 60-67.*
12. Green, N. Some Layout Issues for Multimedia Argument Generation. *AAAI 1999 Fall Symposium on Using Layout for the Generation, Understanding or Retrieval of Documents.* Technical Report FS-99-04, 47-51.
13. Carenini, G. and Moore, J. A Strategy for Generating Evaluative Arguments. *Proceedings of the 1st International Conference on Natural Language Generation* (INLG-00).
14. Green, N. , Carenini, G., Kerpedjiev, S., Roth, S., and Moore, J. A Media-Independent Content Language for Integrated Text and Graphics. *Coling-ACL'98 Workshop on Content Visualisation and Intermedia Representation*, J. Pustejovsky & M. Maybury eds.
15. Corio, M. and Lapalme, G. Integrated generation of graphics and text: a corpus study. *Coling-ACL'98 Workshop on Content Visualisation and Intermedia Representation*, J. Pustejovsky & M. Maybry eds.
16. Energy Information Administration, U.S. Department of Energy. *Greenhouse Gases, Global Climate Change, and Energy.* Available at http://www.eia.doe.gov/oiaf/1605/ggccebro/chapter1.html .
17. Cornell Lab of Ornithology, Ithaca, New York. *Birdscope*.
18. Humphrey, W.S. *A Discipline for Software Engineering*. Addison-Wesley, 1995, pp. 274-283.
19. Paraboni, I., and van Deemter, K. Issues for the Generation of Document Deixis. In André et al. (Eds.), *Deixis, Demonstration and Deictic Belief in Multimedia Contexts, Proceedings of the Workshop associated with the 11th European Summer School in Logic, Language and Information (ESSLLI)*, Utrecht, The Netherlands, 1999, pp. 43-48.
20. Mann, W.C., and Thompson, S.A. Rhetorical Structure Theory: Toward a functional theory of text organization. *Text* 8(3) (1988), 243-281.
21. Mayberry, K.J., and Golden, R.E. *For Argument's Sake: A Guide to Writing Effective Arguments.* Second edition. HarperCollins College Publishers, New York, 1996.

Exploiting Uncertainty and Incomplete Knowledge in Deceptive Argumentation

Valeria Carofiglio and Fiorella de Rosis

Department of Informatics, University of Bari, Italy
{carofiglio, derosis}@di.uniba.it
WWW home page: http://aos2.uniba.it:8080/IntInt.html

1 Introduction

Argumentation is not always sincere. This is evident in competitive domains like politics or trading but occurs, as well, in domains in which debates are commonly considered to be governed by 'purely rational', non malicious goals and forms of reasoning, like science [3,4,5,7]. When people argue for or against a claim, the differences of arguments they employ are due, in part, to differences in the data they know or in the importance they attach to each argument. However, these differences may originate, as well, from the use of arguments that are not fully sincere: it is therefore worth reflecting on whether and how the various forms of deception that may be employed in argumentation might be formalised.
In this paper, we distinguish between 'uncertain' and 'incomplete' knowledge and we prove how both of them might be exploited by the arguing agent to introduce several forms of deception in argumentation, with a different balance between 'impact' and 'safety' of the deception attempt.

2 Argumentation Forms

Let us start from the classical Toulmin's example that is described in http://www.mtsu.edu/mkrueger/toulmin.html): see Table 1. Argumentation may employ purely logical arguments, as in Table 2. However, in the majority of cases (as in Table 1), uncertainty is introduced in the qualifier (and therefore the warrant), the backing of warrant or the influence of the rebuttal.

Various systems have been proposed and prototyped, to show how logical and uncertain argumentation may be formalised: belief networks, in particular, proved to be a powerful tool to model uncertain argumentation, as they enable representing the various warrants that concur to supporting some claim from different data, each with its weight and by appropriately considering the dependency among the data when they exist [10]. Once a model has been built, it may be employed to reason about the effect of evidence available on the claim or, in a sort of 'hypothetical' reasoning, to assess which data combinations produce a desired impact on the claim. For instance, in BIAS, arguments and rebuttals are generated from two belief networks, representing the arguing Agent's first and second order beliefs in the domain [7].

V.N. Alexandrov et al. (Eds.): ICCS 2001, LNCS 2073, pp. 1019–1028, 2001.
© Springer-Verlag Berlin Heidelberg 2001

Table 1. An example from Stephen Toulmin: The uses of Arguments [11].

Since Russia has violated 50 of 52 intl agreements.	DATA
Therefore, probably	QUALIFIER
Russia would violate the proposed ban on nuclear weapon testing,	CLAIM
Since	
Past violations are symptomatic of probable future violations	WARRANT
Unless	
The ban on nuclear weapons testing	REBUTTAL
is significantly different from the violated agreements.	
Expert X states that nations that have been chronic violators	BAKING OF WARRANT
nearly always continue such acts.	

Table 2. Logical argumentation.

DATA	Russia has violated *all* intl agreements in the past.
QUALIFIER	Therefore, certainly,
CLAIM	Russia would violate the proposed ban on nuclear weapon testing,
REBUTTAL	The ban on nuclear weapons testing is significantly different from the violated agreements.
WARRANT	Past violations are always symptomatic of future violations
BACKING OF WARRANT	Expert Y states that nations that have been chronic violators always continue such acts.

The limit of belief networks is, though, in the amount of knowledge required to model the reasoning process: the probability of data have to be known a priori, as well as the links between data and claim, that must be expressed in terms of conditional probability distributions: these parameters may be evaluated subjectively or learned from a training set. The problem of what to do when this knowledge is incomplete, that is, of how argumentation in conditions of 'incomplete knowledge' may be modelled, is still open. We try to imagine, in Table 3, how Toulmin's example would be modified, in this case. The three examined cases (argumentation in a situation of logical, uncertain and incomplete knowledge) are summarised in Table 4: in this table, we employ the following ***Notations:***

- *x denotes a nation*
- *y denotes a member of the set of intl agreements Y*
- *R denotes 'Russia'*
- *N denotes 'the proposed ban on nuclear weapon testing'*
- *$D(x, y)$: "Nation x violated intl agreement y in the past"*
- *$\forall y D(R, y)$: "Russia violated all intl agreements y in the past"*
- *$C(x, y)$: "Nation x will violate the intl agreement y in the future"*
- *$C(R, N)$: "Russia will violate the proposed ban on nuclear weapon testing N in the future"*
- *$W(x, y)$: "Past violations of intl agreements by any nation x imply future violations of intl agreements y in the set Y": $\forall x \forall y (D(x, y) \rightarrow C(x, y))$*
- *$B(e, x, y)$: "Expert e says that past violations of intl agreements by any nation x imply future violations of intl agreements y": $Say(E, W(x, y))$*
- *$R(y, N)$: "The ban on nuclear weapon testing N is significantly different from the violated agreements":$(N \notin Y)$*
- *All the values $(v_1, v_2, ...v_h, ...v_k, ...)$ denote numbers in the (0, 1) interval.*

In uncertain argumentation (third column), the warrant's qualifier provides a measure of the impact that a considered data produces on the claim; an argument

Table 3. Argumentation in a situation of incomplete knowledge.

DATA	We don't know exactly how *frequently* Russia violated intl agreements.
CLAIM	It *is rather credible* that Russia would violate the proposed ban on nuclear weapon testing, but it is also plausible that it will not. We can only make an interval estimate for this event.
WARRANT	*To some extent*, past violations *are* symptomatic of future violations. However, they may be not. There is a number of cases in which we don't know which was the relationship between the two events.
BACKING OF WARRANT	Expert Z states that a large proportion of the nations that had such a record of violations continued such actions.However, there are also nations that did not. And, unfortunately, there are, as well, several cases in which Z can't say whether violations occurred or not.

is said to be "good" if it induces a desired probability value on the claim. This desired value may be obtained by focusing on a data that influences the claim in the desired direction or by introducing some rebuttal that, in a way, strenghtens or reduces the impact of the considered warrant. It should be noticed that warrants, as well as their backings, are not necessarily unique. For instance, several experts might exist, with different viewpoints, that attach different qualifiers to the same warrant; in this case, some criteria has to be applied for deciding which of them should be evoked to achieve the desired effect on the claim. As we said, bayesian reasoning may be applied when all probability parameters attached to the different components of the current argument are known: $P(C(x,y)|D(x,y))$ and $P(C(x,y)|(D(x,y))$. When only incomplete knowledge about these items is available, instead, an interval value for the probability of data and claims may be estimated (fourth column of Table 3). This means that, when an argument is evaluated, a lower and an upper bound for the probability of the claim may be calculated; the convenience of the warrant is linked to the width of this interval and to its lower value: the more the extremes of the interval are near to the probability of the claim the Arguing agent wants to achieve, the more the argument will be effective; the more the interval is large, the more doubt will be left, by the argument, in the Interlocutor's mind.As a consequence, 'uncertain' and 'incomplete-knowledge' argumentation strategies differ, in our opinion, in the way the warrant and its backing are examined, when selecting an 'appropriate' argument for a claim: in uncertain argumentation, to obtain a desired probability value for the claim, the strength of the link between data and claim (the warrant's qualifier) has to be considered.

In incomplete-knowledge argumentation, the type and level of ignorance about this qualifier and its effect on the width of the interval for the probability of the claim have to be considered instead. This opens the possibility of a wide range of enforceable (and, as we will see, even deceptive) argumentation forms.

Table 4. A summary formalisation of logical, uncertain and incomplete-knowledge argumentation forms.

LOGICAL ARG.	UNCERTAIN ARG.	INCOMPLETE ARG.
CLAIM $C(x, y)$		
Goal: $C(R, N)$	**Goal**: $Prob(C(R, N)) = v_g$	**Goal**: $Bel(C(R, N)) = v_1$ $Plau(C(R, N)) = v_2$
DATA $D(x, y)$		
$\forall y D(R, y)$	$\forall y Prob(D(R, y)) = v_1$	$\forall y Prob(D(R, y)) \in [v_3, v_4]$ with $[v_3, v_4] \subseteq [0, 1]$ or $Bel(D(R, y)) = v_3$ and $Plau(D(R, y)) = v_4$
ON WARRANT $W(x, y)$		
$W^l(x, y)$: $\forall x, \forall y$ $((D(x, y) \rightarrow C(x, y)))$ (which , together with $\forall y D(R, y, induces C(R, y))$	$W^u(x, y)$: $\forall x, \forall y$ $Prob(C(x, y) \mid D(x, y)) = v_h$ $Prob(C(x, y) \mid D(x, y)) = v_h$ This together with $Prob(D(R, y)) = v_1$ induces $Prob(C(R, y)) = v_2$ with $v_2 = v_g$	$W^i(x, y)$: $\forall x, \forall y$ $Prob(D(x, y) \rightarrow C(x, y)) = v_h$ $Prob(D(x, y) \rightarrow \neg C(x, y)) = v_k$ $Prob(D(x, y) \rightarrow ?C(x, y)) = v_m$ This together with $Prob(D(R, y)) \in [v_3, v_4]$, induces $Bel(C(R, y)) = v_1$, and $Plau(C(R, y)) = v_2$
ON REBUTTAL $R(y, N)$ with $N \notin Y$		
$\neg Instance - of(y, N)$ (which induces $W^l(x, N) = unknown$	$Prob(C(x, y) \mid D(x, y), R(y, N)) = v_h$ $Prob(C(x, y) \mid \neg D(x, y), R(y, N)) = v_k$ etc for the other combinations; this, together with $Prob(D(R, y)) = v_1$ and $Prob(R(y, N)) = v_i$, induces $Prob(C(R, N)) = v_j = v_g$	
ON BACKING OF WARRANT $B(E, x, y)$		
$\exists e Say(e, W^l(x, y)))$ which together with $Believable(e)$, induces $W^l(x, y)$	$\exists e Say(e, W^u(x, y)))$ which together with $Believable(e)$, induces $W^u(x, y)$	$\exists e Say(e, W^i(x, y)))$ which together with $Believable(e)$, induces $W^i(x, y)$

3 Belief and Plausibility Measures in Dempster&Shafer Theory

To show how Dempster & Shafer's (D&S) theory may be employed in modeling incomplete-knowledge argumentation, we first briefly introduce the main concepts behind this theory (for more details, see [1,9]). We will denote with D a generic data and with C a generic claim connected to D.
Definitions: let $T = \{t_1, ..., t_j, ..., t_n\}$ be the set of mutually exclusive and exhaustive hypotheses we wish to test (the *frame of discernment*, in Dempster&Shafer' s terms); in our scenario, as we deal with boolean variables, $T = \{C, \neg C\}$; let Θ be the set of all subsets of T; again, in our scenario: $\Theta = \{C, \neg C, ?C\}$, where $?C$ denotes '$C or \neg C$'. Let $S = \{s_1, ..., s_i, ..., s_m\}$ be a set of possible answers to 'a question related to T'; in our case, for instance:

- s_1: $D \rightarrow C$ denotes the proposition: 'An evidence about the data D implies an uncertain evidence about the claim C'
- s_2: $D \rightarrow \neg C$ denotes the proposition: 'An evidence about the data D implies an uncertain evidence about the claim $\neg C$'
- s_3: $D \rightarrow ?C$ denotes the proposition: 'An evidence about the data D implies an uncertain evidence about the claim C'

We write '$s_i C t_j$' to denote that s_i and t_j are 'compatible'.
We start from a probability distribution P over S, defined according to some subjective or objective evaluation method; in the example: $P(s_1)$, $P(s_2)$ and $P(s_3)$. The meaning of these probability values may be seen as follows: "*As far as I know, data D implies claim C, with a probability* $P(s_1)$*; it implies* $\neg C$ *with a probability* $P(s_2)$*; in the rest of the cases, I cannot say anything about the relation between D and C* ".
The distribution of probability on the elements of S enables us to compute a belief function $Bel(\Theta_h)$ on subsets of T, as follows:

- $Bel(\Theta_h)$=P$\{s_i \mid if\ s_i C t_j\ then\ t_j\ is\ in\ \Theta_h\ \}$.

$Bel(\Theta_h)$ is a measure of the belief we commit to Θ_h, based on P. This belief function assigns, in particular, a belief value to all elements t_j of our set of hypotheses T. It enables, as well, computing the Dempster&Shafer's plausibility of these elements as follows: $Plau(t_j) = 1 - Bel(\neg t_j)$. This upper bound is a measure of how much belief we commit to $T\ \Theta_h$.
The range of variation of the probability we may attach to t_j varies in the interval $[Bel(t_j), Plau(t_j)]$; the width of this interval is a measure of the 'level of doubt' in our beliefs. As a consequence, if $Bel(t_j) = Plau(t_j) = c$ we may say that we believe t_j to a degree c and $\neg t_j$ to a degree $(1-c)$, that our degree of uncertainty about t_j is equal to $Bel(t_j)$while our doubt about this uncertainty value is null. D&S's belief combination rule may be applied to combine incomplete knowledge from different information sources, as well; that is, in argumentation, to estimate the degree of belief of elements in T when several warrents on the same claim exist.

4 How to Deceive?

Deceptive argumentation differs from sincere argumentation in the assumption that A may cite facts that do not correspond to his own beliefs. This entails, to A, the risk of being discovered, by I, in a deception attempt, that he needs to avoid. Therefore, in deceptional argumentation, the convenience of an argument depends, at the same time, on its *impact* on the claim and on its *safety* (probability of not being discovered by I in a deception attempt). Deception may be applied to a combination of data, warrant, its backing and rebuttal. The selection of the deception form depends of what A considers to be the most 'convenient' means to achieve his goal about the claim.
In another paper, we analyse the forms of deception that may be applied when

uncertainty is measured in bayesian terms [6].In this short paper, we will focus our discussion on a comparison between bayesian and D&S reasoning. We wish to prove, on one side, that uncertainty may serve to achieve the (possibly deceptive) argumentation goal in several ways and, on the other side, that, by deceptively simulating incomplete knowledge, the safety of the deception attempt may be increased, though the impact of the argument is reduced.

4.1 Deception in Uncertain Reasoning

The example in Table 1 might be formalised by assigning the following parameters to the warrant:

Example 1:
$Prob(C(x,y)|D(x,y)) = .9999$;
$Prob(C(x,y)|D(x,y)) = .01$;
In this case, if $P(D(R,y)) = .96$, then $Prob(C(R,y)) = .96$
If, instead, $D(R,y)$ is false, the probability of $C(R,y)$ lowers to .01

Let us now assume that A believes that I is uncertain about the claim (that is, that $Prob(C(R,y)) = .5$ in I's mind). Let us suppose, as well, that A believes that this claim is true, but that his goal is to keep I in doubt about it: that is, he wants to obtain, through his argumentation, that the probability that I assigns to the claim remains invaried (= .5).

Deception form 1: manipulation of uncertainty parameters.
If A reasons in bayesian terms, he may deceptively operate on the uncertainty of the warrant, by pretending, in his conversation with I, that the conditional probability values are such, that the data have a very low impact on the claim:

For instance, A might say:
"There is no relation between past and future violations of intl agreements"
$(Prob(C(x,y)|D(x,y)) = Prob(C(x,y)|(D(x,y)) = .5)$;
Alternatively, he might operate directly on uncertainty of the data:
"Russia violated about half of the agreements, in the past" $(P(D(R,y)) = .5)$.

In both cases, he will obtain that $P(C(R,y)) = .5$. Notice that, due to the 'extreme' values of conditional probabilities, example 1 is not much different from a case of 'purely logical' reasoning: deception form 1 may therefore be considered as a form of deception that consists in introducing uncertainty in logical argumentation, with the advantage of having the possibility to deceive without lying.

Deception form 2: introduction of a rebuttal.
A might evoke, as well, a rebuttal $R(y,N)$, by manipulating the conditional probability table so that the impact of the combination of $D(R,y)$ and $R(y,N)$ on $C(R,y)$ is close to 0.

A might, for instance, say:
"*Nothing may be said about the probability that a nation will violate a particular future*

agreement N, if it violated the previous intl agreements but N is significantly different from these agreements; on the contrary, if N is similar to them, it is very likely that this nation will violate it as well; finally, the probability of a violation is very low, when agreements have not been violated in the past".

The previous sentence equates to setting the following conditional probability distribution for the mentioned events:

$Prob(C(x,N)|D(x,y),R(y,N)) = .5$

$Prob(C(x,N)|D(x,y),\neg R(y,N)) = .9999$

$Prob(C(x,N)|(D(x,y),R(y,N)) = .01$

$Prob(C(x,N)|(D(x,y),\neg R(y,N)) = .01$

Notice that this table is compatible with the values in Example 1. Notice also that, in this case, if $R(y,N)$ and $D(x,y)$ are both true, their combined effect is that the probability of $C(x,N)$ is still equal to .5. Therefore, the declaration that "*It is true that Russia violated intl agreements in the past; however, the ban on nuclear weapon testing is significantly different from the violated agreements*" might help A to leave I in a state of uncertainty about $C(x,N)$, while accepting the data $D(x,y)$ as true.

4.2 Deception in Incomplete-Knowledge Reasoning

We now consider the case in which only incomplete knowledge about the relationship between data and claim is available.

Example 2: backing of warrant by Expert E1

Expert E1 says the following:

"*In 80 % of cases, it was observed that, when a nation was known to systematically violate intl agreements in the past, it continued doing the same*": $P_1(D(x,y) \rightarrow C(x,y)) = .8$

"*It almost never happened that a nation which was known to have systematically violated intl agreements in the past stopped doing such action* ": $P_1(D(x,y) \rightarrow (C(x,y)) = .01$

"*There is a 20 % of cases in which the relation between past and future violations is unknown*": $P_1(D(x,y) \rightarrow ?C(x,y)) = .19$, where $?C(x,y) = C(x,y) or \neg C(x,y)$.

Notice that the difference between Examples 1 and 2 is that, in the second case, knowledge about the effect of the data on the claim is incomplete. According to D&S's theory, if $D(x,y)$ is true, the probability of $C(x,y)$ varies in the (belief, plausibility) interval $(.8,1)$, while the probability of $\neg C(x,y)$ varies in the interval $(0,.2)$. Notice that the belief and the plausibility of $C(x,y)$ are both high: the width of the interval between the two values is not very large, due to the presence of a single backing of warrant and of a not too incomplete knowledge.

Translation into argument: "*Let us suppose that Russia violated all intl agreements in the past. The degree of belief in the fact that Russia will violate future agreements, based on the present state of knowledge, is equal to .8. However, the plausibility of this fact is equal 1: this means that the degree of belief of this claim might go up to 1, should all information that is not available at present be so as to confirm the warrant*".

Deception form 3: simulation of incomplete knowledge.

Let us suppose that A has a complete, although uncertain, knowledge about the

domain in question: that is, he may estimate, either subjectively or objectively, all uncertainty parameters, as in Example 1. If we compare Example 2 with Example 1, we may notice that A might *simulate* an incomplete knowledge to increase, in I's mind, the level of doubt about the claim: he might indirectly deceive I about the claim by deceiving her, in fact, about his own level of knowledge about the warrant. In fact, if $D(x, y)$ is true, $P(C(x, y)) = .99$ in the first case, while, in the second case, it lies in the interval $(.8, 1)$.

Deception form 4: introduction of an 'uninformed' information source.

Let us now assume that A introduces some different backing about the same warrant, for instance by citing the opinion of an Expert E2 whose knowledge is different and more incomplete than knowledge of E1. In this example, the probabilities of evidences might be settled as follows: the relationship between data and claim is available.

Example 3

Expert E2 says the following:

"*In 30 % of the cases, when a nation was known to systematically violate intl agreements, it continued doing the same*": $Say(E2, P_2(D(x, y) \rightarrow C(x, y)) = .3)$.

"*It never happened that a nation which was known to have systematically violated intl agreements in the past stopped doing such action* ": $Say(E2, P_2(D(x, y) \rightarrow \neg C(x, y)) = 0)$.

"*There is a 70 % of cases in which the relation between past and future violations is unknown*": $Say(E2, P_2(D(x, y) \rightarrow ?C(x, y)) = .7)$.

This knowledge leads to a $(.3, 1)$ interval estimate for the probability of the claim. We already saw how the first interval estimate could be interpreted. In a similar way, the interpretation of this second interval is the following: "*If Expert E2 was the only information source, in the present state of knowledge the claim could not be believed, as this information has a low degree of belief*". Notice that in this case, while the degree of belief in the claim is low, the width of the interval is very large, due to the high degree of incompleteness in knowledge of E2. Evoking this expert rather than expert E1 might be employed deceptively, by A, to induce a high level of doubt in I's mind. Notice also that, as the plausibility of the claim is very high, the risk, to A, of being discovered, by I, in a deception attempt is rather controlled: if discovered, A might always say: "*Sorry, I didn't know!*",or "*Sorry, E2 said it!*": this way, he will appear less guilty than if he had to confess a lie.

Deception form 5: introduction of a 'confounding' backing.

As a last example, let us now look at how the two separate backings of warrants in examples 2 and 3 may be integrated with D&S's rule of combination of evidences and how this integration may be exploited deceitfully. According to this rule, if both E1 and E2 are cited and $D(x, y)$ is true, the range of variation for the degree of belief in $C(x, y)$ varies in the interval $(.63, .1)$: we will not go into mathematical details. Notice that, in this case, the degree of belief is lower than in the case in which only knowledge of E1 was available, while nothing changed about the plausibility. The effect of introducing, in the argumentation,

expert E2's viewpoint is that the range of doubt about the claim (the width of the belief-plausibility interval) increased. This example shows that, to insinuate, in the Interlocutor's mind, a doubt about the claim, the arguing Agent may deceptively introduce the viewpoint of another Expert: even in the presence of a relatively informed previous opinion, if the second expert's knowledge is limited or wrong, the final effect will be to 'confound' the interlocutor. This application of D&S's combination rule appears to simulate in a rather effective way some particular types of deceptive argumentation, in which confounding arguments are introduced to divert the interlocutor's attention from the truth, to insinuate doubt in her mind or to increase her level of ignorance.

5 Final Considerations

This short paper is a first contribution to a reflection on how uncertain and incomplete knowledge might be considered in simulating deceptive argumentation. We may conclude that logical, uncertain and incomplete-knowledge reasoning define distinct scenarios: in these scenarios, deceptional arguments decrease of impact on the desired goal about the claim, with the advantage of reducing, at the same time, the negative consequences of being discovered, by the interlocutor, in a deceitful attempt (or, at least, with the advantage of keeping some opportunity of defending oneself). Suspicion in humans is, with no doubt, inherently uncertain: in human communities, it is very uncommon to be absolutely confident in other people's claims; so, as deception implies placing oneself on the interlocutor's viewpoint to reason about the effect of a deceiving attempt, it is reasonable to assume that deception should be simulated in conditions of uncertainty. Situations of 'incomplete knowledge' are very frequent as well, both among humans and in MultiAgent systems, consistently with the idea of an 'open world' with a limited possibility of observing the other Agents' behaviour and of situations whose occurrence and whose relationship with the rest of the world are known only in part. The interest of D&S's theory is that it enables to come to a decision, in these situations of incomplete knowledge. The possibility to induce an interval estimate of the degree of belief in the claim by simulating or exaggerating his ignorance leaves, to the arguing agent, the opportunity of considering, in his decision, what he presumes to be the interlocutor's "personality". 'Pessimistic' or 'optimistic' interlocutors will look at the same (belief, plausibility) interval with different viewpoints:"*I will not get convinced about this claim until I'll have more evidence*" vs "*I accept it until there is some couterevidence*". One might argue that D&S theory does not correspond to a realistic way of modeling human reasoning: this is certainly true. As Tversky and Kahneman proved in their seminal paper, humans tend to apply ad hoc heuristics rather than probability theory, in taking their decisions [12]. A still more curious finding is reported by Barwise, who found that even mathematicians do not necessarily reason logically about mathematical question issues such as 'There are quite a few prime numbers' [2]. So, it's absolutely unlikely that humans apply, in incomplete knowledge reasoning, D&S's theory. However, this type of objection

relates to the very long debate on whether computerised decision-making should try to emulate human reasoning as it is or should apply its own reasoning style, to try to come to a 'correct' decision. We tend to support the second solution and tend to prefer mathematically correct and well grounded theories to 'ad hoc' heuristics, like, for instance, those that have been proposed, for years, to handle uncertainty in 'expert systems' ad have now been abandoned: but this is and will be, of course, a matter of debate for still longtime.

Acknowledgements
We are indebted with Cristiano Castelfranchi for involving us in his long-lasting interest towards deception, for his high-value suggestions and critiques to our work and, in particular, for his hints about the role of uncertainty in this, intriguing indeed, form of reasoning.

References

1. Barnett, J. A.: "Calculating Dempster-Shafer Plausibility". IEEE Transactions on Pattern Analysis and Machine Intelligence. 13, 6, 1991.
2. Barwise, J: "Monotone quantifiers and admissible sets". In J E Fenstad, R O Gandy and G E Sacks: Generalized Recursion Theory. North Holland Publ Co, 1978.
3. Buller,D. B., Burgoon, J. K., Buslig, A., and Roiger, J.: "Testing interpersonal deception theory; the language of interpersonal deception". Communication theory, 6, 3, 1996.
4. Castelfranchi, C. and Poggi, I.: Lying as pretending to give information. Pretending to Communicate, H. Parret (Ed), Springer Verlag, 1993.
5. Castelfranchi, C., Poggi, I.: Bugie, finzioni e sotterfugi. Carocci Publ Co, 1998.
6. de Rosis, F., Castelfranchi, C., Carofiglio, V.: Can computers deliberately deceive? A simulation attempt of Turing's Imitation Game. Sumbitted for publication.
7. Jitnah, N., Zukerman, I., McConachy, R. and George, S.,(2000): Towards the Generation of Rebuttals in a Bayesian Argumentation System. In INLG'2000 Proceedings – the First International Natural Language Generation Conference, pp. 39-46, Mitzpe Ramon, Israel.
8. Kashy, D. and DePaulo, B.: "Who lies?" Journal of Personality and Social Psychology. 70, 5, 1996.
9. Shafer, G. and Logan, R.: Implementing Dempster's Rule for Hierarchical Evidence. Artificial Intelligence, 33, 1987.
10. Spiegelhalter, D. J.: "Probabilistic Reasoning in Predictive Expert Systems". Uncertainty in Artificial Intelligence. L. N. Kanal and J. F. Lemmer (Eds), Elsevier Science Publishers B. V. (North Holland), vol. 4, 1986.
11. Toulmin, S.: "The use of argument", Cambridge University Press, Cambridge MA, 1958.
12. Tversky, A. and Kahneman, D.: "Causal schemata in judgments under uncertainty". In progress in social psychology, ed. M. Fishbein. Hillsdale, N.J.: Lawrence Erlbaum, 1977.
13. Zlotkin, G. and Rosenschein, J.: "Incomplete information and deception in multiagent negotiation". XII International Joint Conference on Artificial Intelligence, Sydney, 1991.

Integrating Computation into the Physics Curriculum*

Harvey Gould[1] and Jan Tobochnik[2]

[1] Department of Physics, Clark University, Worcester, MA 01610-1477, USA
http://physics.clarku.edu/~hgould
[2] Department of Physics, Kalamazoo College, Kalamazoo, MI 49006-3295, USA
http://cc.kzoo.edu/~jant

Abstract. Some of the challenges of incorporating computational methods into the physics curriculum are discussed. These challenges include teaching the methods of computational physics in the same spirit as we presently teach mathematics, changing the curriculum to reflect the new ways of thinking that have arisen due to the use of computer technology, and making use of the Web to make the process of teaching more like that of research.

1 Introduction

Every discipline has its own language that helps its practitioners communicate ideas efficiently with one another. The goal of much of education is to empower students to learn and use that language. Every discipline also uses and contributes ideas and techniques from other disciplines. For example, physicists use mathematics and computational methods developed by practitioners in other disciplines. Ideally, students take courses in many disciplines and can utilize ideas and skills they learn in one discipline and apply them to another. For example, biologists expect their students to learn chemistry from the chemists, and engineers expect their students to learn physics from physicists.

Because computation has become an important way of doing physics, it is important that physics students learn how to program and use computers effectively. Should we expect that students will learn how to program from computer scientists? How can we integrate computational methods into the physics curriculum?

2 An Analogy to Mathematics

The existing physics curricula already provides models for incorporating the methods of another discipline, mathematics, into physics education. It is useful to review the various approaches that have been adopted and analyze how well these approaches have worked.

* Supported in part by National Science Foundation grant PHY-9801878.

V.N. Alexandrov et al. (Eds.): ICCS 2001, LNCS 2073, pp. 1031–1040, 2001.
© Springer-Verlag Berlin Heidelberg 2001

The usual approach is to require physics students to take a certain number of courses in the mathematics department. For example, introductory physics courses are frequently divided into those that use algebra and those that use calculus, and students in the latter courses usually take calculus as a co-requisite. Upper level physics courses usually have prerequisites such as linear algebra, vector calculus, and differential equations. How well does this model work? In general, the answer is not very well. A common complaint is that students learn very little from their math courses that is useful for physics. Sometimes it is argued that math courses spend too much time proving abstract theorems rather than teaching the mathematical tools that physicists need. However, our experience has been that even when this argument does not apply, there still is little carry over. The reason seems to be that even if students learn to use a skill in one context, it is difficult for them to use the same skill in a different context. Skills taught out of context are rarely learned very well. We note that the new accrediting requirements for undergraduate engineering education are based on the same conclusion [1]. Courses in physics are no longer required. Instead engineering departments can incorporate physics into their own courses. Most engineering departments have not taken this approach, at least not in the short term, because of the need for more staff, the large effort in time and resources necessary to change their courses, and the political obstacles.

Should physics departments drop the mathematics course requirements for a physics major and teach the mathematics ourselves? We suggest that if this approach were adopted, our students would be better able to use mathematics in physics and other application areas as well. Of course, we believe that students can learn important concepts from mathematicians. Students should take math courses for their intrinsic interest that are taught by mathematicians as if all the students in the course were math majors. Physics and other science students would then become more familiar with the language of mathematics, would understand the foundations underlying the methods they use in their physics courses, and would learn new ways of thinking that might give them a better understanding of physical phenomena. It would also allow physicists to communicate with mathematicians.

On a smaller scale the debate on how to introduce mathematics is repeated in many physics courses, particularly standard theoretical courses such as classical mechanics, electrodynamics, and quantum mechanics. For example, some textbooks and courses begin with a discussion of mathematical techniques. Other texts and courses integrate the mathematics throughout as needed. Even the same author might take both approaches. Griffiths has written two excellent and popular undergraduate texts. In his electrodynamics text he begins with a chapter on vector calculus [2]. However, in his quantum mechanics text he starts with the physics and introduces the solution of partial differential equations, ideas of probability, delta functions, and linear algebra in later chapters as they are needed [3]. We conclude that the best approach either depends on the material being covered or is a matter of taste.

Physics departments are aware of the difficulties that students have using the skills they learned in mathematics courses in their physics courses. These difficulties have led many physics departments to offer courses in mathematical methods for physicists. In this approach only those mathematical techniques that are useful for physicists are discussed, and the number and rigor of the mathematical proofs are reduced. The techniques are usually discussed with specific applications in mind. There is the obvious advantage of efficiency in this approach, but one limitation is that the physics context is not usually fully discussed because of lack of time. For various reasons, the majority of physics departments have not taken this approach.

A related issue is that many upper division physics courses have become very theoretical. For example, Griffiths' Quantum Mechanics text contains only one graph of experimental data. Admittedly, the text does discuss experimental data in the text and problems, but the presence of only one graph is telling. The lack of integration of theoretical and experimental physics is also a problem in the physics curriculum. The relative lack of experimental physics in the curriculum is driven in part by the fact that modern experimental equipment is very expensive and that creating good laboratory experiments is very time consuming.

The issues related to the integration of mathematics and experimental physics into the physics curriculum are relevant to a discussion of how to integrate computation. However, we have the advantage that we are doing something relatively new, and for the most part we do not need to undo an already existing set of courses. On the other hand, creating a whole new approach can be daunting.

3 Computer Simulations

We look to how the computer is used in physics research to determine what is important to include in physics education [4]. Many theoretical problems require numerical calculations such as the evaluation of an integral, obtaining the roots of an equation, and matrix manipulations. In some cases, such as diagrammatic calculations, symbolic manipulation is essential for keeping track of the different classes of diagrams. Computers are essential for the control of experiments and the collection and analysis of data, and are used in this context to implement statistical procedures for extracting a signal from noise, do fourier transforms to obtain power spectra, fit data to functional forms, and plot data in a way that helps to interpret its meaning.

These uses of the computer in scientific research can be viewed as applications of specific tools, much like an oscilloscope is used to make voltage measurements or a mathematical transformation is used to simplify a mathematical expression. In these cases we believe it makes sense to integrate the use of the tool with the physical application. Although there are numerical methods courses taught in mathematics departments and some computational physics courses focus on numerical analysis, we believe that such courses are not as effective for physics students as would integrating the specific tools directly into physics courses.

However, there is another way of using computers, known as *computer simulation*, that allows us to do physics in a fundamentally different way [5]. One distinction between computer simulation and numerical analysis is that in the latter way, the user does as much as possible analytically before giving the problem to a computer. In some ways, doing numerical analysis on a computer is an extension of techniques used before computers were invented. For example, in the past a human calculator would generate tables of sines and cosines; now a computer can generate these values very quickly. However, there are ways of doing numerical analysis that were inconceivable before the advent of computers. For example, the generation of high-order Feynman diagrams and the evaluation of the corresponding integrals using computational graph theory and Monte Carlo methods. The distinction between numerical analysis and simulation is one of emphasis.

Computer simulation is a way of doing physics that is distinct from the way physics was done before the advent of computers. In a computer simulation we might use various numerical methods and even some symbolic manipulations. However, the style and motivation is very different and is analogous to a laboratory experiment. For this reason computer simulations are sometimes referred to as computer experiments. We first develop a model that can be represented by an algorithm. The model plays a role analogous to the physical system of interest in an experimental system. Sometimes these models are formulated in ways that are related to traditional ways of doing theoretical physics. For example, a molecular dynamics simulation involves solving Newton's equations of motion for a large collection of particles. In other cases, the models are designed with the nature of digital computers in mind. Examples include cellular automata models of fluids and traffic flow.

Just as we need to calibrate a measurement apparatus, we have to test our program and compare its outcome with known results in limiting cases. Then data is collected and finally, the data is displayed and analyzed. In the real world, our initial data frequently leads to further improvements in the program and more data collection.

Computer simulations also have some features that are distinct from laboratory experiments. For example, in experimental physics each new kind of measurement requires a new piece of equipment and may preclude other measurements. In a computer simulation a new measurement requires only some additional code within the existing program. Simulations also allow us to determine quantities that cannot be measured in a laboratory experiment. Of course, simulations are limited by computer resources that limit the size of the system studied and the time that it can be simulated. In many cases these limitations are severe.

One advantage of analytical solutions is that a solution frequently can be written in terms of a parameter so that more than one case is readily available. In contrast, separate simulations frequently must be done for each value of the parameters of interest. However, an advantage of simulations is that many modifications of the model require simple changes in the program (for example,

changing the force law in a molecular dynamics simulation), whereas even minor changes in a theoretical model can make an analytical calculation impossible. Analytical calculations usually require approximations whose consequences are not known. Simulations frequently use numerical procedures that are exact in principle, although the results are approximate due to statistical errors and limitations due to the effects of finite time and size. However, in many cases the nature of these limitations is not known.

Computer simulations have become an important part of research in physics and are increasing in importance in other fields of science. This importance is reason enough to incorporate them in the undergraduate curriculum. More importantly, computer simulations provide the easiest way of involving physics students in the process of scientific research. Writing and running simulations includes many of the aspects of scientific research such as model generation, testing, analyzing data, interpreting data, and drawing general conclusions. And the flexibility of simulations means that they can be done at many different levels for just about any field of physics as well as many fields outside of physics. Hence, we have much freedom in determining where in the curriculum to add computer simulations.

4 Integrating Computer Simulations into the Curriculum

We now discuss ways of integrating computer simulations into the physics curriculum. As we have mentioned, we believe that most other uses of the computer should be integrated directly into existing courses and discussed as needed. We will find that the same issues arise as we discussed earlier in the context of integrating mathematics. We will divide our discussion into the introductory curricula and upper division curricula.

Until recently, most attempts to integrate programming and simulations into introductory level physics courses have been unsuccessful. There are too many demands on students at this level. Most students still have difficulty with basic calculus concepts and many are not even adept in algebraic manipulations. In addition, we know from research in physics education that there are many fundamental concepts that students have much difficulty understanding. They come to physics with a view of the world that frequently is inconsistent with the physicist's world view. At the introductory level we must use Occam's razor and include only those uses of the computer that can enhance conceptual understanding. We cannot expect to teach computational science skills such as programming, algorithm design, and other ingredients that go into designing a computer simulation.

However, there is a noteworthy exception that is being developed by Ruth Chabay and Bruce Sherwood [6]. They are using the programming language Python in their introductory physics courses and are having students write sophisticated computer simulations with powerful three-dimensional graphics with only a couple of hours of instruction [7]. One of the keys to their success is that the graphics statements are largely hidden from the users who can add graphical

objects to their programs with very little effort. Nevertheless, many of the ways of thinking that we would want to introduce when using computer simulations are present. So far, this work has been done with a relatively sophisticated group of students and experienced instructors at Carnegie-Mellon University, and it remains to be seen how large a population of introductory students can be handled in this way.

Another successful approach is the use of *physlets* as pioneered by Wolfgang Christian and his collaborators [8]. Physlets are Java applets built into Web pages using Javascript. The advantages of this approach include that there is a common user interface, the ability of instructors to tailor the physlets to their own needs, and a good set of questions that have been developed to go along with existing physlets. Students can use physlets to make plots and animations and collect data. The disadvantage of this approach is that in most cases students do not learn what is behind the simulation. However, we believe that physlets can help student learning as well as introduce students to the possibilities of simulations.

A commercial product, Interactive Physics, it useful for learning about mechanics [9]. Students can create various kinds of objects and forces between objects, and can collect data and draw graphs. Although many problems have been written for introductory texts using Interactive Physics, most of them are animated versions of traditional textbook problems and do not help the student learn much about more realistic situations nor understand why systems behave the way they do.

There are many other simulation programs available (for example through Physics Academic Software) [10]. However, none of these programs have as their goal the teaching of computational physics.

To summarize the situation for introductory physics courses, we believe that at present the only realistic goal for most instructors is to use already written simulations to help give students a visual and dynamic representation of the physical systems that they are studying. In this context it is very difficult to teach the kinds of thinking that goes into designing a computer simulation.

There are more possibilities in the upper level courses. We believe all physics students should learn how to do computer simulations. This involvement is probably the only realistic way for many undergraduate students to engage in an activity analogous to actual physics research within the context of a course. Laboratory experiments are frequently too costly and time consuming to have students plan an experiment, get the apparatus to work, collect data, and analyze the results. Such work is possible in a separate junior/senior lab, but is too difficult to do in most courses. The analogous process of doing theoretical research is in general too difficult for undergraduates. However, computers are inexpensive, readily available, and fast, so that every student can have access to a physics research tool.

One of the obstacles is learning a programming language well. For over fifteen years we have advocated that students' first programming language should have a simple and clean syntax that incorporates easy to use graphics statements

and is platform independent. We began with True Basic, which has worked very well and serves as an excellent introduction to Fortran, C, and other procedural languages. However, because of the advantages of object-oriented programming and the popularity of Java, we are now using and recommending Java. In collaboration with Wolfgang Christian of Davidson College, we currently are writing the third edition of our computer simulation text in Java [11].

How can students learn a programming language? This issue is similar to the questions we raised for mathematics. Although, physics students frequently take courses from the computer science department, our experience is that the introductory programming courses are not very effective. Also many issues that are important in simulations are of much less interest to computer scientists. More importantly, it takes too many computer science courses before students become sufficiently proficient to write computer simulations on their own.

How can we expect students to learn programming while they are learning physics? The answer is that we can focus on those parts of the language that are useful for doing computer simulations, and we can provide templates and other utilities that the student can use. Also, because there is a context that provides a meaningful reason to write a program, physics students have a higher motivation to write a program that works and does something useful. In fact, most of us learned how to program without taking any courses, just as every child learns how to speak without taking a course in public speaking. Just as we urge students to take mathematics courses to learn mathematics on its own terms, we should urge students to do the same for computer science. There are ways of thinking in computer science that provide a foundation to the tools we are using and that will become more important in computational science in the future. However, we should not rely on computer science courses as a prerequisite for learning to do computer simulations.

We are still confronted with the question of whether we should teach a separate computer simulation course or integrate computer simulations into other physics courses. In most cases we believe that physics departments should do both. The motivation for a separate course is much the same as for a junior/senior course on mathematical methods or a junior/senior laboratory. However, because of the nature of computer simulations, it is possible to teach a meaningful course on computer simulation methods and applications to even first-year students. Once the students have the basic skills, they can utilize these skills in other courses. We stress that for such a course to have maximum impact on the students and the physics curriculum, the emphasis of the course should be on computer simulations rather than numerical analysis.

Unfortunately, even when a separate course is offered, it has not yet led to much use of computer simulations in other courses in most physics departments. The reasons for this lack of impact are that the course is rarely required of all majors, faculty teaching other courses are unfamiliar with simulations or do not have the time to change their courses, and the lack of readily available resources for incorporating simulations. We are trying to rectify this last obstacle

for thermal and statistical physics courses by developing applets and various Java utilities and templates and other curricular materials [12].

Some textbooks in classical mechanics have included a few computer exercises for which the programming is minimized. The focus of these exercises is usually similar to the standard problems and does not allow students the opportunity to engage in more open-ended possibilities that are analogous to research problems. However, it might be possible to add a computer simulation lab to a course or several courses so that students can learn enough programming to write a simulation. The advantage is that their work can be done in a specific physical context such as modeling particle motion in mechanics or wave packet propagation in quantum mechanics.

So far much of our discussion has focused on using the computer in standard physics contexts. Such contexts can be very useful. For example, much of the interesting behavior in particle dynamics such as chaotic phenomena is very difficult to analyze without a computer. The motion of wave packets in a potential cannot be easily visualized without a computer. These are obvious applications that should be done in the relevant physics courses, and we can design open-ended problems that will allow students the opportunity to explore interesting physics as well as obtain a better understanding of traditional concepts.

However, real change in the curriculum will come when the use of computer simulations pushes us to broaden the focus of the physics curriculum. For example, we can extend the simulation of particle systems to the simulation of dynamical systems in general. Why not expand simulations of statistical mechanics models using Monte Carlo and molecular dynamics methods to include more general studies of complex systems such as traffic flow, epidemiology, and neural networks? In both cases we see that the extension of computer simulations beyond traditional topics in physics leads naturally to what we call computational science. Many computational approaches such as genetic algorithms and cellular automata have their origins outside of physics. Including these approaches would provide a powerful pathway to understanding complex systems through computational science. Physics courses may be the most natural setting for introducing computational science because physics historically has been at the forefront in developing new ways of solving problems experimentally, theoretically, and now computationally.

5 Making Teaching Count

The use of the Internet as a vehicle for delivering curricular materials has many implications for teaching. In particular, it allows us to share our course materials and our approach with instructors at other institutions. One reason that teaching is not taken as seriously as research is that our teaching reputations are local and the quality is not easily evaluated. The advent of the Web has already started to change this situation.

We can also use the Web to develop curricular materials in a way that takes advantage of the collective work of many people. We are inspired by the freely

available Linux operating system, which was inspired by the vision of a single individual, but which has been developed by substantial contributions from many people. Anyone is free to work on any additional features or improvements and all new code is easily available for others to use and critique. At the same time, there is always a stable release available that incorporates new features only after they have been thoroughly tested. We are trying to follow this example and develop a core of curricular materials for statistical and thermal physics. It is too soon to say, but we almost have a sufficient core of material that would be useful to other instructors, and without any advertising on our part, other scientists are beginning to make suggestions for improvements. It remains to be seen if others will contribute substantially to this curriculum development project, but the possibility exists for "open source" curriculum development projects in physics and other areas.

Acknowledgements

The authors would like to thank Wolfgang Christian and Joshua Gould for their patience while teaching us much about Java and for their development of the Java templates and utilities that we believe will make Java programming accessible to a much wider group of physics instructors and their students. We also acknowledge the National Science Foundation for its support.

References

1. More information about the Accreditation Board for Engineering and Technology can be found at `http://www.abet.org/`.
2. D. J. Griffiths, *Introduction to Electrodynamics*, 3rd Edition, Prentice Hall (1999).
3. D. J. Griffiths, *Introduction to Quantum Mechanics*, Prentice Hall (1995).
4. This section is based on J. Tobochnik and H. Gould, "Teaching Computational Physics to Undergraduates," D. Stauffer, ed., *Annual Reviews of Computational Physics IX*, World-Scientific Press (to be published).
5. See for example, K. Binder in *Thermodynamics and Statistical Physics: Teaching Modern Physics*, pp. 45–66, Proc. 4th IUPAP Teaching Modern Physics Conf., M. G. Velarde and F. Cuadros, eds., World Scientific Press (1995).
6. R. Chabay and B. Sherwood, *Matter & Interactions*, John Wiley & Sons, to be published. More information about the text and their course at Carnegie-Mellon University is available at `http://cil.andrew.cmu.edu/mi.html`.
7. More information about the three-dimensional graphics module for Python is available in D. Scherer, P. Dubois, and B. Sherwood, "VPython: 3D interactive scientific graphics for students," Comput. Sci. Engin., Sept./Oct. 2000, 82–88 and at `http://cil.andrew.cmu.edu/projects/visual/`.
8. See for example, W. Christian and M. Belloni, *Teaching Physics with Interactive Curricular Material*, Prentice Hall (2001), W. Christian, M. Belloni, and M. Dancy, "Physlets: Java Tools for a Web-Based Physics Curriculum," this conference proceedings, and `http://webphysics.davidson.edu/Applets/Applets.html`.
9. More information about Interactive Physics can be found at `http://www.workingmodel.com/products/ip.html`.

10. Information about Physics Academic Software can be found at `http://webassign.net/pasnew/`.
11. H. Gould, J. Tobochnik, and W. Christian, *Introduction to Computer Simulation Methods*, third edition (unpublished). A preliminary version of the third edition will be available this summer from the Simulations in Physics Web site, `http://sip.clarku.edu`. The second edition by H. Gould and J. Tobochnik, Addison-Wesley (1996) is still available.
12. For information about the Statistical and Thermal Physics (STP) curriculum development project, see `http://stp.clarku.edu`.

Musical Acoustics and Computational Science

N. Giordano[1] and J. Roberts[1,2]

[1] Department of Physics, Purdue University, West Lafayette, IN 47907, USA,
ng@physics.purdue.edu,
WWW home page: http://www.physics.purdue.edu/~ng
[2] Current address: Oberlin College, Oberlin OH 44074, USA

Abstract. There are many interesting problems in musical acoustics that can only be dealt with via computational methods. For the most part, the essential physics is readily accessible at an introductory level, making this an excellent source of examples and research projects for undergraduate students. This theme is illustrated with a quick tour of the issues encountered in modeling the guitar.

1 Introduction

Computers have been associated with music for many decades. This association has been quite extensive, and has included the analysis and composition of music, as well as the synthesis of musical tones. The synthesis problem has been attacked in several ways; computers can be used to make sounds that cannot be produced by any known musical instrument, and they can be used to mimic specific instruments so as to perform compositions that could not be played by a single performer with a conventional instrument (e.g., a twelve voice fugue). In this paper we explore one particular aspect of the synthesis problem: the construction of a musical tone.

There are several different ways in which a computer can mimic the tones produced by a particular instrument. To a first approximation, most musical tones are composed of a collection of harmonic waveforms, so it is possible to assemble an approximate musical tone by combining sinusoids according to various recipes. However, this is not an easy way to obtain tones that sound realistic. The time dependence of the harmonic content, especially during the initial portion of the tone, is extremely difficult to characterize and copy, and the component waveforms are often not precisely harmonic. Another approach is to record typical waveforms from a real instrument and then construct other tones for the instrument (e.g., at a different pitch or loudness level) by algorithmically altering the sampled waveforms. These two approaches, which loosely speaking can be termed additive synthesis and sampling synthesis, are widely used. However, improvements in computer power are now making possible a third approach known as physical modeling. This approach involves computational modeling of the instrument using fundamental physics, i.e., Newton's laws. (Quantum mechanics and relativity do not appear to play significant roles in musical acoustics.) The underlying physics of these simulations is relatively elementary, and many of the

V.N. Alexandrov et al. (Eds.): ICCS 2001, LNCS 2073, pp. 1041–1050, 2001.
© Springer-Verlag Berlin Heidelberg 2001

key concepts, including the wave equation, the vibration of flexible strings, and Fourier analysis, are well within the grasp of undergraduate students. As a result, this field affords many opportunities for interesting, original, and forefront research projects for undergraduates.

Our group has been involved for several years in physical modeling of the piano [1,2]. That work has included simulations of various components of the piano along with experiments to test and refine the simulations. In this paper we describe the beginnings of a similar project for the guitar. Note that there has previously been some interesting and very insightful modeling of the guitar by Richardson and coworkers [3,4,5]. Our approach differs from Richardson's in several ways, the most important being that our calculations are carried out in the time domain (Richardson et al. worked in the frequency domain) which tends to make things more intuitive, and in our opinion this will be a more convenient route to the ultimate goal of a complete and playable a numerical guitar.

2 Overview of the Guitar and Modeling Strategy

Figure 1 contains a schematic drawing of an acoustic guitar showing the major elements of the model. All motion starts with the strings. Here we show only one string; it is secured on the left at a point just beyond the bridge, and on the right at a tuning peg. The portion of the string that is free to vibrate (the "speaking length") extends from the bridge to one of the frets; our figure shows just one fret, but there are actually many spaced along the neck (although at a given moment only one comes into contact with the string). The player controls which fret is in contact with the string, thereby adjusting the speaking length and hence the frequency of vibration. The string is set into motion when it is plucked by the player. This results in a time dependent force on the bridge, that drives the soundboard. The sides and back of the body also vibrate, but they are thicker than the soundboard and so vibrate much less [6]. The motion of the soundboard also causes the air inside the body to move, and the combined soundboard/air resonator has several strongly coupled modes [7]. This motion of the body of the instrument results in the sound that we hear.

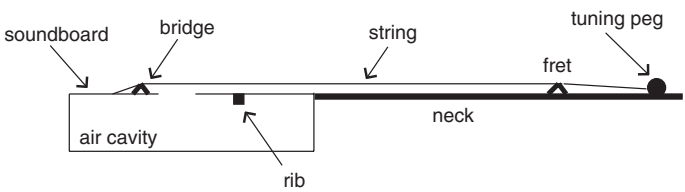


Fig. 1. Very rough schematic of an acoustic guitar (not to scale).

A complete computational model of the guitar must deal with the vibrations of the string, the soundboard, the air in the cavity, and the air surrounding the

body. Such a complete model has not yet been constructed, although Richardson and coworkers [3,4,5] have carried out a detailed analysis of the body vibrations without the air, along with a treatment of the outgoing sound produced by the body alone. In this paper we will describe how one can deal with the string and the soundboard to obtain an (approximate) estimate of the resulting sound pressure. The vibrations of strings and plates are "elementary" problems that are seen in most mechanics courses at the sophomore or junior level. However, we will see that it is necessary to go a bit beyond the simplest descriptions of strings and plates to construct a modestly realistic numerical guitar.

3 The Flexible String: Solving the Wave Equation

The string is perhaps the key element of our simulation, since it is the origin of the driving force for all of the other components of the instrument. The equation of motion for an ideal flexible string is just the usual wave equation

$$\frac{\partial^2 y}{\partial t^2} = \frac{T}{\mu}\frac{\partial^2 y}{\partial x^2} \ , \tag{1}$$

where y is the string displacement, x is position as measured along the string, T is the tension, and μ is the mass per unit length. It is well known that the solutions to this equation are undamped waves that move at a speed $c = \sqrt{T/\mu}$. We have previously described an exact, numerical, time domain solution of this equation [8], so we will not repeat all of the details here (see also [9,10]). The general approach is to discretize both x and t, in steps of Δx and Δt respectively, so that $y(x,t) \to y(i\Delta x, n\Delta t) \to y(i,n)$, where i and n are integers. The partial derivatives in (1) can then be written in terms of $y(i,n)$, $y(i \pm 1, n)$, etc., in the usual way [8,9]. This equation can be rearranged to obtain $y(i, n+1)$ in terms of $y(i,n)$ and $y(i, n-1)$; i.e., the string displacement at spatial location i at the next time step $n+1$ can be calculated from knowledge of the string displacement at previous time steps (n and $n-1$).

This explicit method is very fast, and much more convenient than alternative implicit methods [11]. Moreover, this explicit method is in this case *exact*, provided that the two step sizes are chosen so that $\Delta x/\Delta t = c$. A simple physical interpretation of this point was given in [8].

Some typical results obtained with this computational approach are shown in Fig. 2. The initial string displacement was a perfect "pluck" (i.e., triangle) as shown in the top profile in the figure. Successive plots show how this initial profile breaks up, with kinks traveling left and right. Note also the succession of many small kinks in the central region. These are due discreteness effects in combination with the assumption of a sharp kink at the plucking point. Such a singularity is, of course, unphysical; no string is perfectly flexible and the inevitable stiffness will always round off the kink. Moreover, when a player plucks the string she will always produce a somewhat rounded profile due to the finite size of her finger or pick.

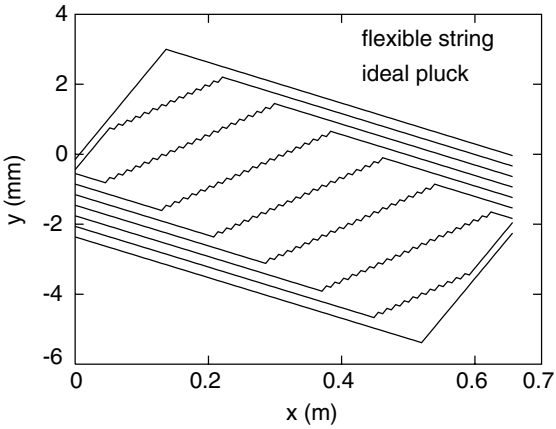


Fig. 2. Snapshots of the string pro le for a flexible string. The string parameters were chosen to match the string B3 of a typical acoustic guitar. The string length was 0.65 m, the diameter 0.24 mm, the density 8000 kg/m^3 (as for steel), and the tension was 149 N (to obtain the desired frequency of ~ 247 Hz). The results show (from top to bottom) the string pro le at $t = 0$, $t = 0.25$ ms, $t = 0.50$ ms, etc., with successive results displaced downwards for clarity. The spatial step size was $\Delta x = 0.65$ mm.

This rounding of the initial string profile can easily be included by simply smoothing off the initial $y(x)$, and the results of a similar calculation with a slightly smoothed initial pluck are shown in Fig. 3. On the scale visible here it does not matter much how one does the smoothing (i.e., with a gaussian convolution or a much simpler smoothing algorithm). As expected, the unphysical difficulty with the kinks is now removed.

4 Adding Stiffness Leads to Anharmonicity

As noted in the previous section, all strings have at least some stiffness, the origin of which can be understood by considering a very thick (large diameter) string. If such a string is bent, there will be a restoring force due to the compression and stretching of the string on the inside and outside of the bend. This force is proportional to the Young's modulus [6,12], and is independent of the tension. When this force is included the equation of motion becomes [6]

$$\frac{\partial^2 y}{\partial t^2} = \frac{T}{\mu}\frac{\partial^2 y}{\partial x^2} - \frac{\pi E r^4}{4\mu}\frac{\partial^4 y}{\partial x^4} \ , \tag{2}$$

where E is the Young's modulus, r is the radius of the wire, and we have assumed a circular cross-section.

For a typical guitar string the stiffness term is small, but it has some important consequences. Physically the stiffness makes the string dispersive, as the

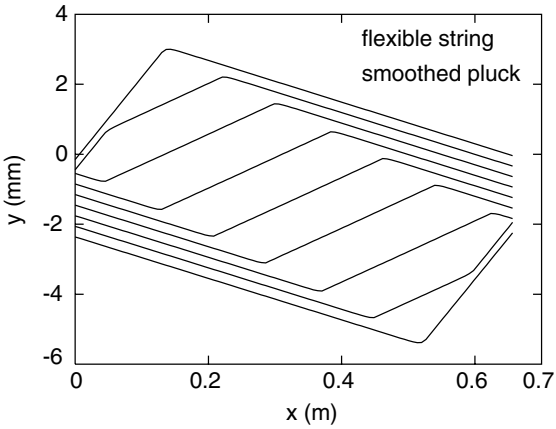


Fig. 3. Snapshots of the string pro le for a flexible string, with the initial plucked pro le smoothed out as explained in the text. The rest of the calculation was the same as in Fig. 2.

wave speed now depends on the frequency. As we will see in a moment, this will have a small but very important effect on the spectrum and the nature of the guitar tone. Computationally the stiffness term affects the accuracy and stability of our time domain/finite difference algorithm. As we have discussed elsewhere [9], it is no longer possible to choose the ratio of the time steps as $\Delta x/\Delta t = c$; doing so would lead to an unstable (divergent) algorithm due to the presence of vibrations that move faster than c. Choosing the ratio of the step sizes to be $r \times c$ with $r > 1$ yields a stable solution if r is sufficiently large (just how large depends on the size of the stiffness term in (2)). While this yields a stable solution, the algorithm is now no longer exact [11]. Fortunately the algorithmic errors are very small for the cases we will be considering; in a plot like Fig. 3 this error is essentially unresolvable.

Let us now return to the physical consequences of the stiffness term. The dispersion that it introduces means that our string is no longer a perfectly harmonic vibrator. For a flexible string the normal modes follow the well known pattern $f_n = nf_1$ where f_1 is the fundamental frequency. For our stiff string the modal frequencies become $f_n = nf_1 + \beta n^3$, where β is proportional to E (note that f_1 is also slightly different from the value for a flexible string) [6]. Hence the modes are no longer harmonically related, but the higher modes are shifted systematically to higher frequencies. This is the source of the effect known as "octave stretching" in pianos [6,9].

The magnitude of this anharmonicity can be seen from the spectrum of the sound produced by our string. Actually, our model is not yet complete enough to calculate the sound directly, since we have not yet considered how to compute the sound pressure. We will therefore rely on the simple but reasonably accu-

rate observation that the sound pressure produced by a vibrating soundboard is approximately proportional to the velocity of the bridge. We have verified this by experiments and calculations for the piano [13,2], and expect it to also be a reasonable approximation for the guitar. However, in order for us to calculate the sound in this manner we must have a model of the soundboard and bridge. We now consider an extremely crude model; we will return to this problem and give a better description of the soundboard in the next section.

The mechanical impedance Z of an object is defined as

$$Z \equiv \frac{F}{v} \ , \tag{3}$$

where F is the applied force and v the velocity of the object. Usually one considers a harmonic force (so that v is at the same frequency); then Z is in general frequency dependent and complex. We have shown through experimental and computational studies that to a first approximation one may treat the impedance of a soundboard as a frequency independent (and real) constant [1,14]. In this case (3) then also becomes the time domain equation of motion of the bridge and soundboard. Independent calculations (to be discussed below) and experiments [15,6,7] give an approximate value of $Z = 100$ kg/s for a guitar soundboard.

Combining this soundboard description (3) with our string enables us to calculate the sound produced by our guitar. The soundboard position is obtained by integrating its velocity as derived from (3) using a finite difference (Euler) method. The force on the soundboard is just the product of the string tension and the slope of the string at the end attached to the bridge. This end moves with the bridge, coupling energy out of the string.

Some results for the sound are given in Fig. 4 which shows the spectra for both a flexible and a stiff steel guitar string. On this scale the two spectra are indistinguishable. The relative amplitudes of the different modes can be understood in terms of the initial plucked profile [9,6].

In order to better show the effect of the stiffness, Fig. 5 shows a greatly expanded view of the spectrum in the neighborhood of the 12th "harmonic" (it may be properly termed a harmonic only for the flexible string; more generally these modes are termed "partials"). As promised, the frequency of this partial for the stiff string is shifted slightly above that of the flexible string. The frequency difference is only about 1 Hz, and is thus not large. However, the effect is noticeable to a listener [6].

5 A Slightly Realistic Soundboard

While the effect of string stiffness can be observed from spectral analysis, it is extremely useful to evaluate the calculated tones by simply listening to them. Unfortunately this is not possible with the printed version of this paper, but can be easily accommodated by visiting our www site (see the URL given above). Such listening tests reveal (in our opinion) that the tones produced by our model are surprisingly good considering the very simple way in which the soundboard

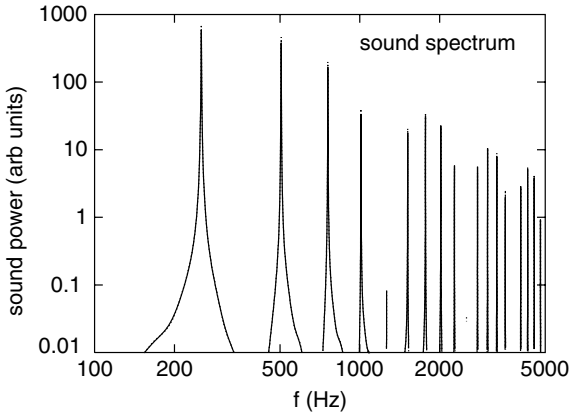


Fig. 4. Sound spectra from the simulations for a flexible (solid curve) and a stiff string (dotted). The two results are indistinguishable on this scale. The simulation parameters were the same as those given in connection with Fig. 2.

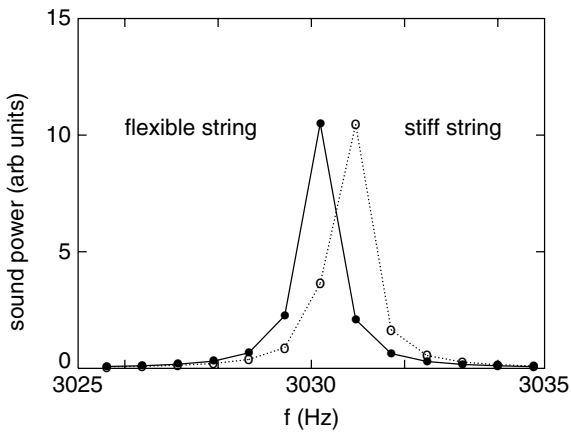


Fig. 5. Sound spectra for a flexible (solid curve) and a stiff string (dotted) from Fig. 4 on a greatly expanded scale. The spectra were estimated with a fast Fourier transform, and the symbols indicate the resolution of the FFT.

was included and the sound pressure derived. Perhaps the most serious weakness of these calculated tones is with the attack; the initial portion of the tone is too "dull" as it seems to turn on instantaneously [16]. This is not surprising, since the soundboard model (3) is effectively massless. That is, it has no inertia so it will respond instantly, with no delay, to the force from the string. A real soundboard will have inertia, and this will cause the soundboard motion and also the sound pressure to build up gradually during the initial portion of the tone.

Let us therefore consider how a more realistic soundboard can be incorporated into our guitar model. The equation of motion of a thin (sound)board is [12,1]

$$\rho h \frac{\partial^2 z}{\partial t^2} = -D_x \frac{\partial^4 z}{\partial x^4} - (D_x \nu_y + D_y \nu_x + 4D_{xy}) \frac{\partial^4 z}{\partial x^2 \partial y^2} - D_y \frac{\partial^4 z}{\partial y^4} + F_s(x, y) \ , \quad (4)$$

where z is the displacement of board, the plate lies in the $x - y$ plane (not to be confused with the variables x and y associated with the string), D_x, D_y, and D_{xy} are stiffness factors that are functions of the Young's moduli (and which are anisotropic), ν_x and ν_y are Poisson's ratios, h is the thickness of the board, ρ is its density, and $F_s(x, y)$ is proportional to the force of the string on the bridge.

This equation can be attacked in the same manner as (1) with a explicit finite difference approach, the details of which have been given elsewhere in the context of the piano [1,2]. A real guitar soundboard has ribs which add stiffness to the board in selected regions, and one rib is shown schematically in Fig. 1. Our soundboard model includes several ribs and also contains a damping term to account for energy loss within the board. Space does not permit more discussion of this soundboard model here, but details of similar calculations are given elsewhere [1,2].

Some results for the sound pressure, again assuming that the sound pressure is proportional to the velocity of the bridge, are given in Fig. 6. Here we show the early part of the sound waveform, i.e., the "attack," for both the crude soundboard (3) and the improved model (4). The two results are quite different, even though the spectra (not shown here) are actually rather similar. The differences are due to two sources. The mechanical impedance that results from (4) is complex and frequency dependent (although its average value is close to $Z = 100$ kg/s). In addition, this soundboard has inertia so it takes some time for the sound waveform to reach an approximately steady state form; here we see that this takes several periods, or around 20 ms. For the crude soundboard there is no inertia and the steady state is reached immediately.

6 Future Directions

The best way to evaluate the success of these modeling efforts is to listen to the calculated tones. A visit to our www site will allow the reader to judge for herself. In our opinion the results are quite encouraging; it is certainly possible to mistake the calculated tones for a real guitar.

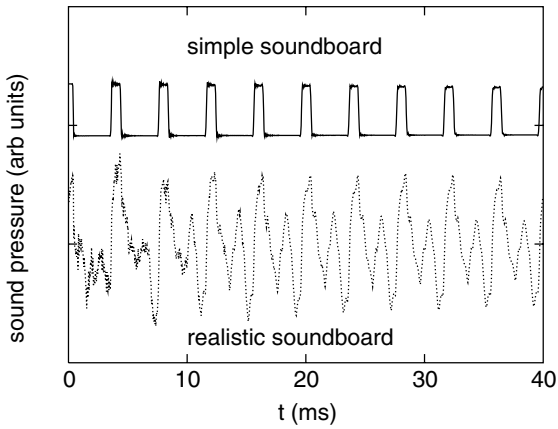


Fig. 6. Early portion of the sound pressure calculated from the simplest soundboard model (3) (solid curve), and from the more realistic model (4) (dotted curve). The simulation parameters were the same as those given in connection with Fig. 2.

Even so, there is much that could and should be added to make this a more realistic calculation. Modeling the entire body of the instrument, including the air cavity, is needed; experiments have shown that the lowest few modes of the air in the cavity and the soundboard mix strongly at frequencies of a few hundred Hz [7]. A direct calculation of the sound pressure in the surrounding air is also needed. Such a calculation has been accomplished in the frequency domain for an anechoic room (i.e., a room with reflectionless walls) [4,5], but a time domain calculation in a more realistic environment would clearly be of interest, and seems quite feasible.

Nearly all of the basic physics involved in this project is at the level of sophomore or junior level mechanics, and hence these modeling projects can be readily undertaken by undergraduate students (and professors!). Calculations of this type are also at the forefront of current research in musical acoustics, as witnessed by the increasing interest in physical modeling of musical instruments [3,17,18,19,20,21,2].

Acknowledgements

We are indebted to B. Martin, T. D. Rossing, and G. Weinreich for their patience in teaching us much about musical acoustics. We thank H. A. Conklin and B. E. Richardson for helpful correspondence, and P. Muzikar for many useful discussions. This work was supported by NSF grant PHY-9988562.

References

1. N. Giordano, "Simple model of a piano soundboard," J. Acoust. Soc. Am. **102**, 1159 (1997).
2. N. Giordano, M. Jiang, and S. Dietz, "Physical modeling of the piano: Design of the model and first results," submitted to J. Acoust. Soc. Am.
3. B. E. Richardson, G. P. Walker, and M. Brooke, "Synthesis of guitar tones from fundamental parameters relating to construction," Proceedings of the Inst. of Acoustics **12**, 757 (1990).
4. M. Brooke and B. E. Richardson, "Numerical modeling of guitar radiation fields using boundary elements," J. Acoust. Soc. Am. **89**, 1878 (1991).
5. B. E. Richardson and M. Brooke, "Modes of vibration and radiation fields of guitars," Proc. Inst. Acoust. **15(3)**, 686 (1993).
6. N. H. Fletcher and T. D. Rossing, *The Physics of Musical Instruments,* (Springer-Verlag, New York, 1991).
7. O. Christensen and B. B. Vistisen, "Simple model for low-frequency guitar function," J. Acoust. Soc. Am. **68**, 758 (1980).
8. N. Giordano, "Physics of vibrating strings," Computers in Physics, March/April, p. 138 (1998).
9. N. Giordano, *Computational Physics,* Prentice-Hall, 1997.
10. We should hasten to add that we are definitely *not* the inventors of this numerical approach to the wave equation.
11. A. Chaigne, "On the use of finite differences for musical instruments. Application to plucked string instruments," J. Acoustique **5**, 181 (1992).
12. S. G. Lekhnitskii, *Anisotropic Plates* (Gordon and Breach, New York, 1968).
13. N. Giordano, "Sound production by a piano soundboard," J. Acoust. Soc. Am. **103**, 1648 (1998).
14. N. Giordano, "Mechanical impedance of a piano soundboard," J. Acoust. Soc. Am. **103**, 2128 (1998).
15. J. Roberts and N. Giordano, unpublished.
16. We should note that our calculated tones all included loss terms in the string equation of motion (2), to model energy loss internal to the string [11,17,18,2].
17. A. Chaigne and A. Askenfelt, "Numerical simulations of piano strings. I. Physical model for a struck string using finite difference methods," J. Acoust. Soc. Am. **95**, 1112 (1994).
18. A. Chaigne and A. Askenfelt, "Numerical simulations of piano strings. II. Comparisons with measurements and systematic exploration of some hammer-string parameters," J. Acoust. Soc. Am. **95**, 1631 (1994).
19. A. Chaigne and V. Doutaut, "Numerical simulations of xylophones. I. Time-domain modeling of the vibrating bars," J. Acoust. Soc. Am. **101**, 539 (1997).
20. V. Doutaut, D. Matignon, and A. Chaigne, "Numerical simulations of xylophones. II. Time-domain modeling of the resonator and of the radiated sound pressure," J. Acoust. Soc. Am. **104**, 1633 (1998).
21. L. Rhaouti, A. Chaigne, and P. Joly, "Time-domain modeling and numerical simulation of a kettledrum," J. Acoust. Soc. Am. **105**, 3545 (1999).

Developing Components and Curricula for a Research-Rich Undergraduate Degree in Computational Physics

Invited paper, 2001 International Conference on Computational Science, San Francisco, May 2001

Rubin H. Landau**

Physics Department, Oregon State University, Corvallis, OR 97331, USA.
rubin@physics.orst.edu, http://www.physics.orst.edu/~rubin

Abstract. A four-year undergraduate curriculum leading to a Bachelor's degree in Computational Physics is described. The courses, texts, and seminars are research- and Web-rich, and culminate in an Advanced Computational Science Laboratory derived from graduate theses and research from NPACI centers and national laboratories. There are important places for Maple, Java, MathML, MatLab, C and Fortran in the curriculum.

1 Overview

We are presently experiencing historically rapid advances in science, technology, and education driven by a dramatic increase in the power and use of computers. Whereas a decade ago computational science educators were content to have undergraduates view scientific computation as "black boxes" and to wait for graduate school for them to learn what is inside the boxes [1], our increasing reliance on computers makes this less true today, and much less true in the future. To adjust to changes in scientific computing, we are developing a four-year, research-rich curriculum leading to Bachelor of Science and Bachleor of Arts degrees in Computational Physics (CP).

Our department has already developed an award-winning undergraduate course in *Computational Physics* [2], a text book [3] that is joining others [4] as proposed models for undergraduate CP courses [5], web-based tutorials and demonstrations that enhance the course and text [6], and a newer-still one-quarter course in *Introductory Scientific Computing* for which we are developing curriculum materials. With the addition of one new course, an *Advanced Computational Science Laboratory*, the modification of one other, and the use of courses offered in other departments, we believe we have assembled a coherent and innovative undergraduate degree program in CP.

** Supported in part by National Science Foundation Grant 9980940 and the National Partnership for Advanced Computational Infrastructure, EOT.

V.N. Alexandrov et al. (Eds.): ICCS 2001, LNCS 2073, pp. 1051–1060, 2001.
© Springer-Verlag Berlin Heidelberg 2001

By teaching some of the computing classes within the Physics Department we will be able to adjust their content and depth to provide a balanced program within the allowed credit limit. This will also permit us to work around the budget difficulties that restrain other departments from teaching shortened versions of the courses taught to their own majors. Our program may well act as a stepping stone for further interdisciplinary programs at the undergraduate and graduate levels.

2 Need for Program

A bachelor's degree in any computation science is rare[7]. To the best of our knowledge there is only one bachelor's degree in CP in the United States [8], and just several physics degrees with minors or specialties in CP [9,10]. Impressively, Trinity College, Dublin has obtained national support to start a B.S. in CP program [11]. We hope that the rarity of degrees in computational science and our promotion will draw new students to our program. In addition, with cooperation across campus, incoming undergraduates will now be presented with a variety of ways to combine computers with science, some in the College of Engineering and some in the College of Science.

Our program is challenging and requires dedicated students. Considering the serious commitment required for interdisciplinary studies, the awarding of a specific degree in CP will, in part, provide recognition for professional schools and employers of the student's extraordinary achievement. This program will also help meet the broadly recognized need to provide undergraduates with research experience, an experience usually associated with highly-ranked universities. Because the research laboratory for computational physics is a virtual world created by the computer, it is easier and quicker to have undergraduates work in this lab than in a "wet" one.

Our goal is to present the job market and graduate schools with undergraduate students possessing competent education and training in physics, applied mathematics, computing, and complex problem-solving. Since the operation of our program also nurtures the communication and team-work skills valued by present employers [12], our students should be valued new hires. We suspect that other physics departments might also follow our model as a good way to revitalize their physics offerings, and that the nation will be served well with these types of graduates.

One need for a program such as ours arises from the documented observations that many of the computer science students now finding jobs do not have the background in mathematics and science needed for technical fields, and that most of the science and engineering students now finding jobs in computer-related fields do not have the requisite background in computation. Another need has been noted by the President's Information Technology Advisory Committee [13] who emphasized the severe shortage of information technology workers. All Computer Science Department throughout the country graduating students at full capacity could not meet this need. The same observation has been made by

the US Department of Commerce and by InfoWeek. PITAC recommends that disciplines in addition to CS supply the workers.

2.1 Educational Objective and Approach

Our objective is to have students understand how to perform scientific computations and experience the interweaving of high-performance computing and communications into the fabric of modern science and engineering. When successful, the mathematical equations and connections among physical idea become alive before students' eyes, and the students understand physical systems at a level usually attained only in a research environment. To do good science and engineering with computers, the student should understand a) how the computer works, b) the relevant science and mathematics, and c) how algorithms and computer simulations are used to connect a) and b).

Much of the computational materials students will encounter in our program come from basic research projects, and will be set in the scientific problem-solving paradigm [2,14]:

$$\boxed{\text{problem} \rightarrow \text{theory} \rightarrow \text{model} \rightarrow \text{implementation}} \leftrightarrow \boxed{\text{assessment}}$$

where the assessment links back to all steps in the big box. This paradigm distinguishes the different steps in scientific problem solving, encourages the use of a variety of tools, and emphasizes the value of continual assessment. Building our material according to this paradigm not only emphasizes the need to know more than what is in the traditional physics curriculum in order to be creative in computational physics, but also makes it easier for students from non-physics disciplines to follow the material and take the courses. While a benefit of our project will be increased understanding of physics content, use of the problem-solving paradigm will deepen scientific process skills [15].

Key components of our program are having students get actively engaged with projects, as if each were an original scientific investigation, and having projects in a large number of areas. In this way students experience the excitement of individual research, get familiar with a large number of approaches, acquire confidence in making a complex system work for them, and continually build upon their accomplishments. We have found this project approach to be flexible and to encourage students to take pride in their work and their creativity. It also works well for independent study or distant learning.

A valuable part of our students' education will be summer involvement with computational research programs within the university and at laboratory and industrial sites. In particular, our program is part of the Education, Outreach, and Training thrust area [16] of the National Partnership for Computational Infrastructure [17], and they are providing us with both support and guidance. They have world-leading supercomputing facilities, training classes, summer workshops, internship programs and collaborations with other NPACI projects, in which we plan to involve our students.

2.2 Computer-Mediated Learning and Accessibility

Our curriculum will be rich with web materials that are used and, at times, developed by the students. This reflects developments in our department and research groups over the last eight years in web-enhanced education [18,19], and our view that the web and computer-mediated instruction will play an increasing role in future scientific computing and education.

While we do not view the web as a good single choice for the teaching medium for general physics and for most students, one cannot beat having a motivated student sit at a computer in a trial-and-error mode in order to learn scientific computing [20]. Further, the web is an ideal environment for computational science: projects are always in a centralized place for students and faculty to observe, codes are there to run or modify, and visualizations can be striking in 3-D, color, sound, and animation. In fact, our planned mix of the web, computer-mediated learning, projects, and lectures for the computational courses is similar to the new pedagogical strategy known as *Just-in-Time Teaching* [21] and the use of *physlets* [22]. Taken together, our approach combines the use of technology and hypermedia to assist learning abstract concepts [23], and the pedagogical strategy known "active learning" or "interactive engagement" [24,25].

The Physics Department has a research group, the Science Accessibility Project [26], that develops ways to make scientific materials accessible to print-disabled (blind and dyslexic) students. In addition, our research group has benefited from a number of academically-gifted students who are seriously dyslexic or physically disabled. Modern computing equipment has helped these students produce high quality research projects and excel in their careers. We will continue to look for ways to use the intellectual and physical leverage provided by computer and communication technology to permit people with disabilities to become productive scientists, and will ensure than our program is open and welcoming to them. Specifically, our program, with NPACI [17] assistance, will incorporate SAP developments as well as assist in SAP research by incorporating the techniques used to produce accessible documents with MathML and XML [27,28] into the course materials.

2.3 Course of Study

In Table 1 we give a sample schedule of the B.S. in CP curriculum. The two-credit courses are parts of the separate *Paradigms* project [29] that has reorganized our department's mid-level undergraduate courses into smaller blocks, with a block covering related ideas normally found in a number of traditional classes. Table 1 is just one possible arrangement of the required courses; others exist, as well as ones in which substitutions are made dependent upon the student's interests and the advisor's consent. The program has 21 credit hours of electives compared to the Physics B.S. degree that has 25 hours. Essentially, we are picking some of the "electives" a student might choose to specialize in computational science.

Table 1. Sample schedule showing proposed curriculum for **B.S. in Computational Physics** with 180 total credits (1 credit = 10 class hours). Computer-intensive courses shown in **bold.** Courses suggested for electives or approved substitution: PH 415, Computer Interfacing; PH 435, Classical Mechanics; MTH 452, Numerical Solution of Ordinary Diffrntl Equations; MTH 453, Numerical Solution of Partial Diffrntl Equations; CS 311, Operating Systems; CS 361, Fundamentals of Software Engineering; PH 428, Rigid Bodies; Ph 441, Physical Optics; PH 481, Thermal and Statistical Phys; PH 621, Classical Dynamics.

	Fall	**Winter**	**Spring**
Fresh (46)	Diffrntl Calculus (MTH 251, 4) Gen Chemistry (CH 201, 3) Fitness/Writing I, 3 Perspective, 3 **CP/CS Seminar** (PH 407, 1)	**Scientific Comptng I** (PH/MTH/CS 265, 3) [or Fall term] Integral Calculus (MTH 252, 4) Gen Chemistry (CH 202, 3) Perspective, 6 [or Fitness/Writing I, 3]	**Intro Computer Sci I** (CS 161, 4) Vector Calculus I (MTH 254, 4) Gen Phys, Rec (PH 211,221; 4,1) Fitness/Writing I, 3 [or Perspective, 3]
Soph (45)	**Intro Computer Sci II** (CS 162, 4) Writing II, 3 Vector Calculus II (MTH 255, 4) Gen Phys, Rec (PH 212,222; 4,1)	**Discrete Math** (MTH 231/235, 3) Infinite Series and Seqncs (MTH 253, 4) Gen Phys, Rec (PH 213,223; 4,1) Perspective, 3	**Scientific Comptng II** (PH 365, 3) **Linear Algebra** (MTH 341, 3) App Diffrntl Eqs (MTH 256, 4) Intro Modern Phys (PH 314, 4)
Jr (44)	**CP Simulations I** (PH 465, 3) **CP Seminar** (PH 407, 1) **Intro Probability** (MTH 361, 3) Oscillations (PH 421, 2) Static Vector Fields (PH 422, 2) Writing III/Speech, 3	**CP Simulations II** (PH 466, 3) **Data Structures** (CS 261, 4) Waves in 1D (PH 424, 2) Quantum Measurement (PH 425, 2) Central Forces (PH 426, 2) Elective/Perspsective, 3	Periodic Systems (PH 427, 2) Class/Quant Mechan (PH 435/451, 3) Energy and Entropy (PH 423, 2) Biology, 4 Perspective/Elective, 3
Sr (45)	**Num. Lin Alg.** (MTH 451, 3) Electromagnetism (PH 431, 3) Mathematical Methods (PH 461, 3) Elective, 6	**Adv CP Lab** (PH 417,517; 3) **Social & Ethical CS** (Synthesis, CS 391, 3) Elective, 6 Synthesis, 3	**Thesis** [**CP Lab+WIC**] (PH 401, 4) **Interact Multi Media** (CS 395, 4) **CP Seminar** (PH 407, 1) Electives, 6

Physics/Mathematics/Computer Science 265, Scientific Computing I

1 Unix, Windows, Maple, Numbers	6 Logical Flow Control
2 Basic Maple, Functions	7 Loops, Numerical Integration
3 Floating Points, Symbolic Computing	8 Complex Arithmetic, Objects
4 Visualization, Calculus, Root Finding	9 OOP, Matrix Computing
5 Classes and Methods	10 General I/O, Applets

An introductory course designed to provide the basic computational tools and techniques needed by lower division students for study in science and engineering. The course is based on a project approach using the problem solving environment *Maple* and the compiled language *Java*. For most students this course will be their first experience with visualization tools, the use of a cluster of workstations sharing a common file system, and the Unix operating system. (Learning Unix is assisted by the web-based *Interactive Unix Tutorial* we have developed and distributed nationally [18].)

While the scientific programming of applications in C and Java is similar, our recent switch to Java in place of C provides an *object-oriented* view towards programming (inclusion of methods with variables), demonstrates the developing potential of platform- and operating-system independent programming, and emphasizes that the web is an integral part of future scientific computing. We have found Java's handling of precision, errors, variable types, and pointers to be superior to C for scientific computing. We also find Java's platform and system independence attractive since this may modify the too rapid (2-3 year) obsolescence of educational software, and encourages distributed computing over the web.

Physics 365, Scientific Computing II

1 Software Basics	6 Diffrntl Equations
2 Errors and Uncertainties	7 Hardware: Memory and CPU
3 Integration & Differentiation	8 Matrix Computing
4 Data Fitting	9 Profiling and Tuning
5 Random Numbers	10 Parallel Computing

An intermediate level course that provides the basic mathematical, numerical, and conceptual elements needed for utilizing computers as virtual scientific laboratories using Java, C, and Fortran. The basics of computer hardware, such as memory and CPU architecture, and shell programming with the Unix operating system are presented. Also studied are the basics of scientific computing: algorithms, precision, efficiency, verification, numerical analysis and associated approximation and round-off errors, algorithm scaling, code profiling, and tuning. Examples are taken from elementary physical systems that make the concepts clear, as well as being easy to compute. The limits of model and algorithm validity is demonstrated by investigating the physics simulation examples in regions for which there is manifest numerical failure.

Physics 407 Computational Physics Seminar Reports of modern happenings, campus research results, and journal articles are presented and discussed. Undergraduates will hear about and learn to think about research topics while advanced students will present results of their projects and research.

Physics 417/517 Advanced Computational Laboratory

Dilos (Giebultowicz)	Monte-Carlo ordering in dilute semiconductors
DFT (Jansen)	Density functional theory of super lattices
DFT2 (Jansen)	Molecular dynamics
Gamow (Landau)	Bound states & resonances of exotic atoms
HF (Jansen)	Hartree-Foch calculations of atoms & molecules
LPOTT (Landau)	K and π elastic scattering from spin 1/2 nuclei
LPOT_{II} (Landau)	MPI code for polarized proton scattering
LPOTp (Landau)	Nucleon-nucleus scattering in momentum space
MD (Rudd, CS)	Molecular dynamics simulations of SiO_2
MEG (Landau)	Principal components of magnetic brain waves
Monte (Giebultowicz)	Monte-Carlo simulations of magnetic thin films
nScatt (Giebultowicz)	Monte-Carlo simulations of neutron diffraction
PiN (Landau)	Quark model of $\pi - N$ interaction
Qflux (Landau)	QCD of 3D quark flux tubes
Shake (Jansen)	Earthquake analysis
Transport (Palmer, NE)	Transport simulations of nuclear storage

We are developing a completely new, advanced computational laboratory in which senior CP students and graduate students will experiment with computer simulations taken from previous M.S. and Ph.D. research projects, as well as from research projects at national laboratories. We are writing and will publish the laboratory manual for this course. The research descriptions and computer simulations will be modified in order to provide a research experience accessible to undergraduate students in a short time (in contrast to the people-years required to develop the research codes originally).

To learn that codes are pieces of scientific literature designed to be read and understood by more than just their authors, the students will run, profile, modify, parallelize, and extend these working codes. The students will run some simulations without knowing what results to expect as a way of learning that codes also function as virtual laboratories built to explore nature. Since the projects will be based on existing research codes, many of the programs will have been written in some version of Fortran. This will be a valuable experience for students since often Fortran is no longer taugh even though the majority of high performance computing applications are presently written in some version of it. In fact, we have heard from some of our industrial colleagues that lack of knowledge of Fortran and lack of experience with running large codes written by others are some of the weakness they find in present graduates.

Physics 465, 466 Computational Physics Simulations

1 Quantum Eigenvalues, Root Finding	9 Quantum Path Integration
2 Anharmonic Oscillations	10 Fractals
3 Fourier Analysis of Oscillations	11 Electrostatic Potentials
4 Unusual Nonlinear Dynamics	12 Heat Flow
5 Differential Chaos in Phase Space	13 Waves on a String
6 Bound States in Momentum Space	14 KdeV Solitons
7 Quantum Scattering, Integral Eqns	15 Sine-Gordon Solitons
8 Thermodynamics: The Ising Model	16 Electronic Wave Packets

The techniques covered in Scientific Computing, PH 265 and 365, are applied and extended to physical problems best attacked with a powerful computer and a compiled language. The problems are taken from realistic systems, with emphasis on subjects not covered in a standard physics curriculum. The students work individually or in teams on projects requiring active learning and analysis.

The course is designed for the student to discuss each project with an instructor and then write it up as an "executive summary" containing: *Problem, Equations, Algorithm, Code listing, Visualization, Discussion, and Critique.* The emphasis is professional, to make a report of the type presented to a boss or manager in a workplace. The goal for the students is to explain just enough to get across that they know what they are talking about, and be certain to convey what they did and their evaluation of the project. As part of the training, the reports are written on the web.

3 Program Evaluation, Student Learning Assessment

The initial and periodic evaluation of our materials will be made by the students in the classes we teach. This will be done regularly through discussions conducted in the Computational Physics Seminar, with web surveys, as well as with the mandatory class evaluation forms. An evaluation of the technical content of our program will be requested from our Advisory Board. Secondary evaluation will be made by the Physics Department of Western Oregon University, who will also enroll interested WOU students in our program. Third-party formative and summative evaluation will be provided by the University of Wisconsin-Madison's *Learning through Evaluation, Adaptation and Dissemination Center* [30], a team that has developed a national reputation for its evaluations of educational reforms that utilize high performance computer technologies, and that aim to recruit and retain women and under represented minorities into the fields of science, mathematics, engineering, and technology.

References

1. *Workshop Report, Undergraduate and Graduate Education in Computational Science*, D. Greenwell, R. Kalia, P. Vashista, and H. Myron, Eds., Louisiana State Univ., Argonne National Lab., April 1991.
2. The Undergraduate Computational Engineering and Sciences (UCES) Project, http://www.krellinst.org/UCES/index.html.
3. R.H. Landau and M. J. Paez (Coauthors), H. Jansen and H. Kowallik (Contributors), *Computational Physics, Problem Solving with Computers*, John Wiley, New York, 1997; http://www.physics.orst.edu/~rubin/CPbook.
4. Introduction to Computer Simulation Methods, Applications to Physical Systems, Second Edition, Harvey Gould and Jan Tobochnik, Addison-Wesley, Reading, http://sip.clarku.edu/.
5. Harvey Gould and Jan Tobochnik, Amer. J. Phys, **67** (1), January 1999; William H. Press, Phys. Today, p 71, July 1998.

6. Northwest Alliance for Computational Science and Engineering, an NSF Metacenter Regional Alliance centered at Oregon State University, http://www.nacse.org.
7. The results of a February 2001 international survey finds 38 graduate and 27 undergraduate programs in all areas of computational science, http://http://www.physics.orst.edu/ rubin/TALKS/CSE_degrees/ahm2001.html.
8. Degree Sequence in Computational Physics, Illinois State University, http://www.phy.ilstu.edu/CompPhys/CP.html;
9. Syracuse University, Bachelor of Arts degree with an Option Physics and Computation, http://suhep.syr.edu/undergraduate.
10. Rensselaer Bachelor of Science in Applied Physics Curriculum, http://www.rpi.edu/dept/phys/Curricula/currAppPhysComp.html.
11. Computational Physics and Chemistry Degree Courses, Trinity College, Dublin (Ireland), http://www.tcd.ie/Physics/Courses/CCCP/CCCPflyer.html
12. *Skills Used Frequently by Physics Bachelors in Selected Employment Sectors*, American Institute of Physics Education and Employment Statistics Division, (1995).
13. President's Information Technology Advisory Committee, http://www.ccic.gov/ac/.
14. The Shodor Education Foundation, Inc., http://www.shodor.org/.
15. R. Root-Bernstein, Discovering, Random House, New York (1989).
16. Education, Outreach, and Training thrust area of the National Partnership for Computational Infrastructure, http://www.npaci.edu/Outreach.
17. National Partnership for Advanced Computational Infrastructure, http://www.npaci.edu/.
18. The Landau Research Group, NACSE in Physics, Oregon State University, http://nacphy.physics.orst.edu.
19. R.H. Landau, H. Kowallik, and M. J. Paez, *Web-Enhanced Undergraduate Course and Book for Computational Physics*, Computers in Physics, 12, (1998); http://www.aip.org/cip/pdf/landau.pdf.
20. P. Davis, *How Undergraduates Learn Computer Skills: Results of a Survey and Focus Group*, T.H.E Journal, **26**, 69, April, 1999.
21. *Just-in-Time Teaching: Blending Active Learning with web Technology*, G.M. Novak, E.T. Patterson, A.D. Gavrin, and W. Christian, Prentice Hall, Upper Saddle River, 1999.
22. *Physlets: Teaching Physics with Interactive Curriculum Material*, W. Christian and M. Belloni, Prentice Hall, Upper Saddle River, 2001.
23. C. Dede, M. Salzman, R.B. Loftin, and D. Sprague, *Multisensory Immersion as a Modeling Environment for Learning Complex Scientific Concepts*, Computer Modeling and Simulation in Science Education, eds. N. Roberts, W. Feurzeig, and B. Hunter, Springer-Verlag, New York, 1999.
24. R.R. Hake, *Interactive-engagement vs. traditional methods*, Am. J. Phys., **66**, 64–74 (1998);
T.E. Sutherland and C.C. Bonwell, eds., *Using active learning in college classes; a range of options for faculty*, Jossey-Bass, San Francisco (1996).
25. D. R. Sokoloff, *Using Interactive Lecture Demonstrations to Create an Active Learning Environment*, The Physics Teacher, 35, 340 (1997).
26. Science Access Project, http://dots.physics.orst.edu/.
27. HTML Math Overview, World Wide Wed Consortium, http://www.w3c.org/math.
28. XML Bridges the Gap, Info World Electric, June 1, 1998 20, Issue 22, p.88-90, http://www.infoworld.com/cgi-bin/displayArchive.pl?/98/22/i09-22.88.htm; S. Mace, U. Flohr, R. Dobson, and T. Graham, Weaving a Better Web, Byte, p58, March 1998.

29. Paradigms in Physics Project, http://www.physics.orst.edu/paradigms/.
30. Learning through Evaluation, Adaptation and Dissemination (LEAD) Center, University of Wisconsin, http://www.cae.wisc.edu/ lead/.

Physlets: Java Tools for a Web-Based Physics Curriculum

Wolfgang Christian[†‡], Mario Belloni[§], and Melissa Dancy[¶]

Physics Department, Davidson College, Davidson, NC 28036, USA.
wochristian@davidson.edu, http://webphysics.davidson.edu

Abstract. An approach to developing curricular material that couples a software design philosophy with physics education research (PER) is described. It is based on open Internet standards such as Java, JavaScript, and HTML as well as research into the e ectiveness of computer-based physics instruction.

1 Overview

> *"Good educational software and teacher-support tools, developed with full understanding of principles of learning, have not yet become the norm."* *How People Learn: Brain, Mind, Experience and School* from Committee on Developments in the Science of Learning, National Research Council National Academy Press, 1999.

The impact of instructional software on mainstream physics instruction has, at present, been minimal. At American Association of Physics Teachers (AAPT) meetings in the 1980s, it was common to see participants sharing floppy disks and trading software for the computer-enabled educational reform that everyone knew was sure to come. It didn't, at least not in the form envisioned by the conference participants. Little of the early educational software was adopted by the mainstream teaching community and almost none of it is still being used today. In contrast, printed material from the much earlier post-Sputnik curricular reform movement — the Berkeley Physics series, for instance — is still available and useful to physics educators, although the pedagogy upon which it was based has gone out of fashion. Will this scenario be repeated and are we doomed like the Greek hero Sisyphus to forever push computational physics up the hill of curriculum reform? Can we expect widespread adoption of computation in the current curricular reform initiative? And, if so, what strategies should we adopt to insure that computational-rich curricula being developed today will be adopted and be in widespread use a decade from now?

[†] email: wochristian@davidson.edu
[§] e-mail: mabelloni@davidson.edu
[¶] e-mail: medancy@davidson.edu
[‡] Supported in part by National Science Foundation grant DUE-9752365.

V.N. Alexandrov et al. (Eds.): ICCS 2001, LNCS 2073, pp. 1061–1073, 2001.
©Springer-Verlag Berlin Heidelberg 2001

It is not surprising that many thoughtful teachers are unwilling to invest the time and energy needed to incorporate computational and educational software into their curriculum. Learning the intricacies of a software package is a poor investment of time for instructors, authors, and publishers if the half-life of the textbook publication cycle is longer than the half life of the computer technology. However, the case can now be made that this throwaway cycle for educational software need not repeat itself and that key technologies are available that enable authoring and distribution of curricular material that will withstand the test of time. Commercial applications have, in effect, provided the education community with a rich and flexible set of standards that are likely to prevail. This paper describes an approach based on virtual machines, meta-languages, and open Internet standards that couples a software design philosophy with research into the effectiveness of computer-based instruction.

2 Physlets

Physlets – "Physics applets" – are small, flexible Java applets that can be used in a wide variety of applications [1]. Many other physics-related Java applets are being produced around the world – some of them very useful for education. However, the class of applets that we call "Physlets" has several attributes that make it valuable for educators.

1. Physlets are pedagogically neutral. Physlets can be used as an element of almost any curriculum with almost any teaching style. Because of their dynamic interactivity, Physlets are ideally suited for interactive engagement methods [2–4] such as Just-in-Time Teaching [5], Peer Instruction [6], and Tutorials [7]. In addition, Physlets can also be used as traditional lecture demonstrations and can be given as end-of-chapter homework.
2. Physlets are simple. The graphics are simple; each Physlet problem should be designed to involve only one facet of a phenomenon, and should not incorporate very much in the way of a user interface. This use keeps Physlets relatively small and eases downloading problems over slow network connections, and removes details that could be more distracting than helpful.
3. Physlets are flexible. Both physical and non-physical situations can be created. All Physlets can be set up and controlled with JavaScript, meaning that a dynamics modeler, such as the Animator Physlet, can be used for almost any subject in mechanics with small changes in the JavaScript and not the Java code. Data analysis can be added when needed with a second Physlet using inter-applet communication.
4. Physlets are web-based. They can run on (almost) any platform and be embedded in almost any type of HTML document, whether it be a homework assignment, a personal web site, or an extensive science out-reach site.
5. Physlets encourage collaboration because they are free for noncommercial use at educational institutions. Physlet archives, that is, compressed archives containing compiled Java programs, can be easily downloaded from the

Davidson College WebPhysics server:
`http://webphysics.davidson.edu/applets/applets.html`.

3 Educational Software Design

The Model-View-Control (MVC) design pattern is one of the most successful software architectures ever invented. It is the basis for the Smalltalk programming language and was used extensively in designing user-interface components for the Java 2 platform. It is well suited to the design of interactive curricular material using Physlets. In this design pattern, the model maintains data and provides methods by which that data can change; the control gives the user the ability to interact with the model using input devices such as the keyboard and the mouse; and the view presents a visual representation of some or all of the model's data. Although there is usually one model and one control, there will often be many views.

A compiled object oriented language, such as Java, should be used to implement models and views because these objects are often very complex and computational speed is important. They are best implemented as separate applets. But the control object is different. The control object is accessed infrequently in comparison to a CPU clock cycle and need not be compiled. More importantly, it should be customizable by the curriculum author because it is difficult for a programmer to anticipate every author's needs. A curriculum author would certainly expect to graph relevant physical quantities as an object moves, but it would be difficult for a programmer to anticipate the various combinations that are needed. For example, will an author want to plot position, energy, or force? What if there are dozens of objects? A scripting language such as JavaScript embedded into an HTML document provides an ideal solution to this problem because the author can change the behavior to suit his or her needs.

Another advantage of using JavaScript to control a model and its views is that control can be distributed through the narrative. The ubiquitous HTML anchor tag — the tag that usually takes the reader to another HTML document — can also be used to execute JavaScript. So can HTML buttons, timers, and user-initiated events such as resting a mouse pointer on an image. In contrast, a monolithic Java applet that combines model, view, and control would remain virtually unchanged from one pedagogic context to the next. HTML augmented with JavaScript enables Physlets to share a common user interface that eases the learning curve across pedagogic contexts.

4 Scripting

The pendulum provides a good example of how to use the MVC pattern to create a Physlet-based exercise. The pendulum model consists of a system of three first-order coupled differential equations. The ODE Physlet solves this

model and the solution is passed to the two views shown in Figure 1 using inter-applet communication. The view on the left shows an animation of a red ball at the end of string. The view on the right is a phase space plot.

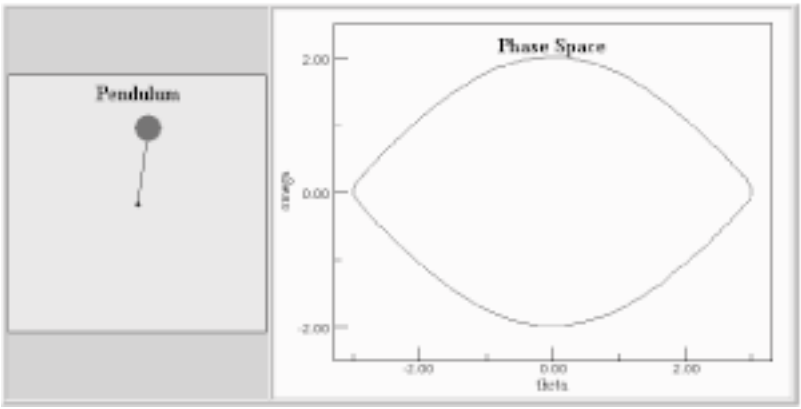


Fig. 1. Pendulum simulation with two data views.

Before an applet can be scripted, it must be properly embedded into an HTML page. Embedding an applet is similar to embedding an image. For example, the DataGraph applet—a typical data-analysis Physlet—is embedded using the following *applet* tag:

```
<applet
    codebase="../classes/"
    archive="DataGraph4_.jar,ODE4_.jar,STools4.jar"
    code="dataGraph.DataGraph.class"
    name="plot_view" width="400"
    height="300">
    <param name="AutoScaleX" value="true">
    <param name="AutoScaleY" value="true">
    <param name="ShowControls" value="true">
</applet>
```

The codebase attribute specifies the directory where the files are located, the archive attribute specifies which files are needed to run the simulation, and the code attribute specifies the object that contains the entry point to the applet. In this case we are running the DataGraph Physlet. This applet can now be referenced by its logical name, *plot_view*, when using JavaScript. Other applets are embedded in a similar fashion except that the ODE applet has zero height and zero width because it does not have an on-screen representation; it merely solves the pendulum's differential equation.

After an applet is embedded, JavaScript is used to invoke its methods using typical object-oriented dot notation starting with the document object as the container. For example, *plot_view* is initialized as follows:

```
document.plot_view.setLabelY("omega");
document.plot_view.setLabelX("theta");
document.plot_view.setTitle("Phase Space");
```

Almost all Physlets support one or more addObject methods designed to instantiate Java objects inside the applet. This method has the following signature:

```
addObject(String name, String attributes);
```

These objects usually, but not always, have an on-screen representation. The first argument is the name of the object to be added, and the second is a comma-delimited list of parameters. An important feature of the addObject method is that it returns a unique integer identifier, id, that can later be used to change properties of the object. For example, the circle shown in Figure 1 was added to the *animation_view* using the following JavaScript statements:

```
ball_id=document.animation_view.addObject(
    "circle", "x=0,y=-1.0,r=10");
document.animation_view.setRGB(ball_id, 255, 0, 0);
```

The addObject method is very forgiving. Parameters can appear in any order and not all parameters need to be specified. Default values are overridden only if the parameter appears in the list. Incorrect and unsupported parameters do not affect the applet and are ignored.

Systems of first-order ordinary differential equations (ODE) can be incorporated into a simulation using the ODE Physlet. For example, the script that models the pendulum simulation can be written as:

```
document.ode.setDefault();
document.ode.addObject(
    "ode","var=t, value=0, rate=1");
document.ode.addObject(
    "ode","var=theta, value=3, rate=omega");
document.ode.addObject(
    "ode", "var=omega, value=0, rate=-sin(theta)");
document.ode.parse();
```

The ODE Physlet must now pass the variable values `t`, `theta`, and `omega` to the two views, the plot and the animation.

Inter-applet communication is performed directly in Java, but the connection must first be established using script. The data-source object in the sending applet must implement the SDataSource interface and the receiving object must implement the SDataListener interface. Both of these objects register their capabilities in a superclass common to all Physlets. JavaScript can then be used

to set up a data connection between the source and the listener using a method with the following signature:

```
makeDataConnection(int sid, int lid, int series,
    String xfunction, String xfunction);
```

The first two parameters, *sid* and *lid*, are integer identifiers for the data source and data listener objects, respectively. The third parameter is a user-defined number that can be used by the data listener to keep track of multiple data sets. The last two parameters are strings representing mathematical functions of any of the data source variables. These functions are evaluated as data passes through the connection so that a single datum, (xfunction, xfunction), is delivered to the listener. For example, the JavaScript necessary to pass the omega and theta variables from the ODE applet to the animation view and the plot view is

```
graph_id=document.graph_view.getGraphID();
ode_id=document.ode.getSourceID();
document.ode.makeDataConnection(
    ode_id,graph_id,1, "theta","omega");
document.ode.makeDataConnection(
    ode_id,ball_id,1, "sin(omega)","cos(omega)");
```

The first two JavaScript statements are necessary to retrieve the integer identifiers for the Java objects that will receive the data. The inter-applet communication mechanism is very fast (because it is implemented in Java) and flexible (because it is implemented in JavaScript). The only change that needs to be made to a script in order to change the data being delivered is to edit the functions, that is, the two string parameters.

5 Curricular Innovation

We have developed close to one thousand Physlet-based problems over the past four years in support of a number of introductory physics texts. A selection of these problems is available on the CD that accompanies the *Physlets* book [1]. More importantly, the Physlets upon which these problems are based are freely distributable for non-commercial educational purposes and are now being adapted to support various curriculum reform initiatives.

It would be foolish to predict the future direction of the computer industry and its impact on education. For example, streaming video is currently a hot technology, and both traditional broadcasters and software companies are competing to establish themselves in this market. However, research has shown that merely watching video has little effect on student learning, and it is unlikely that streaming video will change this result. Small cognitive effects have been shown to occur using video clips if the showing of the clip is accompanied with in-class discussion or if the clip is used for data taking and data analysis [8]. Similarly, database technology has become ubiquitous in our society. It is used to store consumer-shopping profiles for corporate marketing departments

and to manage Christmas card mailing lists at home. But little has been gained in attempts to tailor the curriculum to individual learning styles. Other high-end technologies, such as virtual reality, three-dimensional modelling, and voice recognition, will almost certainly come on-line in the coming decade. However, their most enduring effect on education may be to drive the price/performance ratio of consumer, and hence educational, hardware even lower. These technologies are unlikely to have a significant impact on undergraduate education without a corresponding curricular-development effort and research into their pedagogic effectiveness. Current commercial technologies may, in fact, already be good enough to implement the most effective teaching strategies. Unlike previously written educational software, software written using Internet standards, such as Java and JavaScript, should be accessible for years to come. For computation to have a long lasting impact on science education, it will also need to be based more on successful pedagogy than on the latest compilers, hardware, or algorithms.

5.1 PER: Physics Education Research

Physics Education Research, PER, informs us that technology does not necessarily lead to improved learning and that we are just beginning to understand how it is best used. Two PER researchers, Aaron Titus [9] and Melissa Dancy [10], have used Physlets to study the effect of animation on student assessment and student problem solving ability. Their research focuses on students, not on the Physlets themselves.

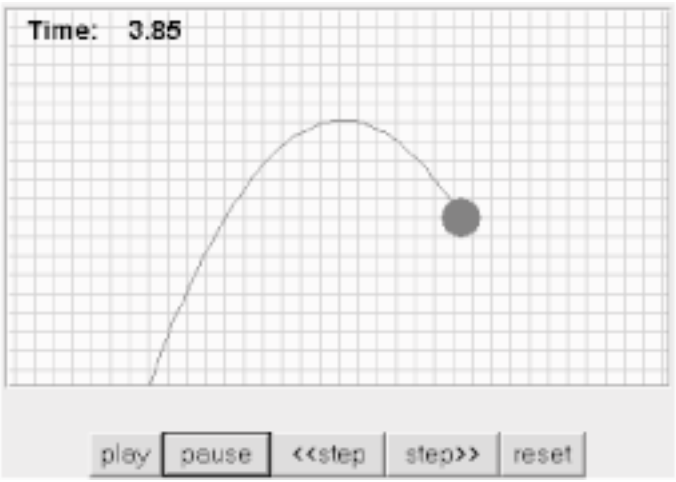


Fig. 2. A media-focused projectile problem.

Titus measured student attitudes and problem-solving approaches while they were solving Physlet-based problems [9]. The study distinguishes between media-enhanced problems where multimedia is used to present what is described in the text, and media-focused problems, where the student must use multimedia elements in the course of solving the problem. Titus found that media-focused problems are fundamentally different from traditional physics problems, and Physlets are ideally suited for these types of problems. Consider an example from kinematics. A traditional projectile problem states the initial velocity and launch angle and asks the student to find the speed at some point in the trajectory. This problem can be media-enhanced by embedding an animation in the text, but this adds little to the value of the problem. Alternatively, this same problem could be a media-focused Physlet problem as shown in Figure 2. In this case, no numbers are given in the text. Instead, the student is asked to find the minimum speed along the trajectory. The student must observe the motion, apply appropriate physics concepts, and make measurements of the parameters he or she deems important within the Physlet. (A mouse-down enables the student to read coordinates.) Only then can the student "solve the problem." Such an approach is remarkably different from typical novice strategies where students attempt to mathematically analyze a problem before qualitatively describing it (an approach often called "plug-and-chug" and characterized by a lack of conceptual thought during the problem-solving process).

Dancy used Physlets to probe students' conceptual understanding by using a standard diagnostic instrument, the Force Concept Inventory [12], in which all thirty static pictures (see Figure 3) were replaced by Physlet-based animations (see Figure 4) [11]. Both quantitative and qualitative data was collected from hundreds of students using the Physlet-based version and the results were statistically analyzed. The study showed that Physlet-based problems are less likely to elicit memorized responses because they allow students to respond to what they see, rather than what they read. Physlets tap into students' intuition and deeply-held misconceptions by eliminating the additional step of translating from words or graphs. In general, students had a better understanding of the intent of the questions when viewing an animation and gave an answer that was more reflective of their actual understanding. Dancy speculates that this may be because the animation looks more like real life than something from a physics textbook.

Both the Titus and the Dancy studies indicate that while computer-based animation can be used for cosmetic and motivational purposes, they are most effective under the following conditions:

1. The animation is integral to the question.
2. The student must interact with the animation to obtain data.

The effectiveness of Physlets likely depends on many factors such as how well the task targets known student difficulties, how students use visual cues given by the Physlet, how important visualization is to the given task, and the appropriateness of the Physlet to the given task. But both studies show that conceptual understanding is key to solving Physlet problems. Without strong conceptual

understanding, students are prone to guess, search for the "right" equation, and lack direction.

Physlet problems generally cannot be correctly solved using "plug-and-chug" methods. The fact that data is not given in the text of the problem requires that students apply proper conceptual understanding to the solution before analyzing data. Therefore, it also seems that Physlet problems may be useful for encouraging a "concept-first" approach to solving problems, where students consider the concepts or principles to be applied to the problem before making calculations.

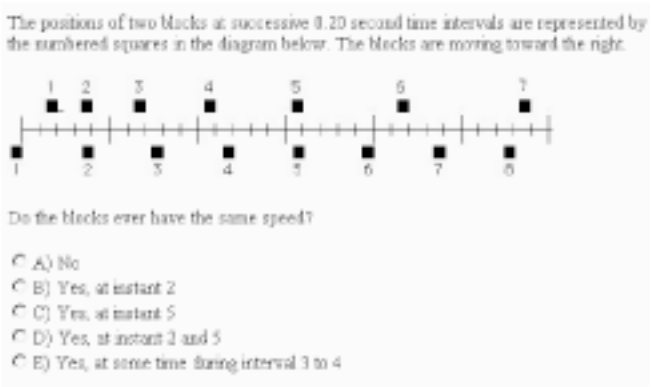


Fig. 3. A text-based Force Concept Inventory question: The positions of two blocks at successive 0.2-second time intervals are represented by the numbered squares in the diagram. The blocks are moving toward the right. Do the blocks ever have the same speed?

5.2 Just-in-Time Teaching

Although the media-rich content and interactivity provided by technology such as Physlets can be pedagogically useful, it can lack the human dimension that is important to effective teaching. Computer Assisted Instruction (CAI) has already been tried on very elaborate proprietary systems. It is unlikely to be improved significantly by being ported to the Internet. To be truly effective, the communication capabilities of the computer must be used to create a feedback loop between instructor and student. A new and particularly promising approach known as Just-in-Time Teaching, JiTT, has been pioneered at Indiana University and the United States Air Force Academy and further developed at Davidson College [5]. It employs a fusion of high-tech and low-tech elements. On the high-tech side, it uses the World Wide Web to deliver multimedia curricular materials and manage electronic communications between faculty and students. On the low-tech side, the approach requires a classroom environment that emphasizes

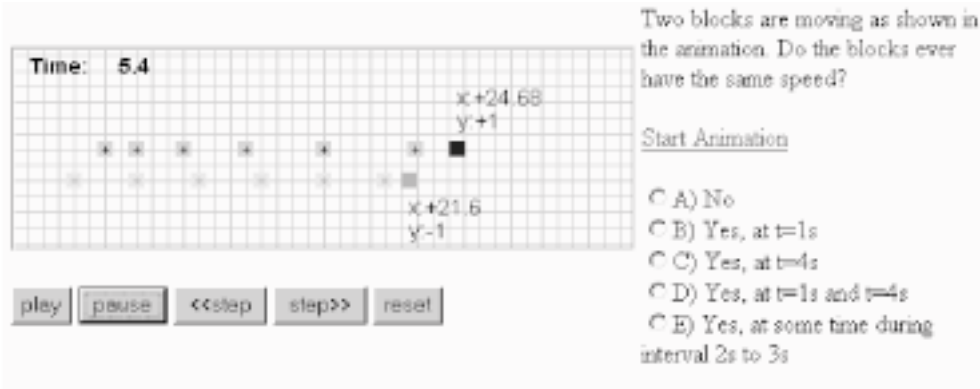


Fig. 4. A Physlet-based Force Concept Inventory question: Two blocks are moving as shown in the animation. Do the blocks ever have the same speed?

personal teacher-student interactions. These disparate elements are combined in several ways, and the interplay produces an educational setting that students find engaging and instructive. The underlying method creates a synergy between the Web and the classroom to increase interactivity and allow rapid response to students' problems.

The JiTT pedagogy exploits an interaction between Web-based study and an active-learner classroom. Essentially, students respond electronically to carefully constructed Web-based assignments, and the instructor reads the student submissions "just-in-time" to adjust the lesson content and activities to suit the students' needs. Thus, the heart of JiTT is the 'feedback loop' formed by the students' outside-of-class preparation which fundamentally affects what happens during the subsequent in-class time. The students come to class prepared and already engaged with the material, and the faculty member already knows where classroom time can best be spent.

Although JiTT can be implemented fully using technically simple Web-based assignments, incorporating Physlet-based questions can heighten the extent to which student understanding can be probed and encouraged. The JiTT strategy as applied in physics education is richer for the incorporation of Physlets. Consider, for example, the puzzles shown in Figures 5 and 6. Students are expected to analyze each situation, apply the relevant physics, and answer specific questions. The faculty member then prepares a lecture in response to the student submissions.

It is interesting to compare the questions. The static Puzzle (Figure 5) involves nearly the same physics as the Physlet-based Puzzle (Figure 6), but what is required of the student in order to solve each puzzle is quite different. In each case, the student must understand the concepts of moment of inertia, torque, angular acceleration, angular velocity, and the relationships between those quantities. In each case, it also behooves the student to draw free body diagrams to

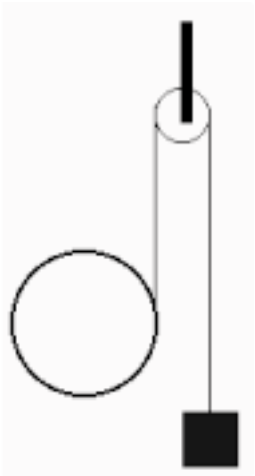


Fig. 5. JiTT Yo-Yo Puzzle Question: Make yourself a yo-yo by wrapping a ne string around a thin hoop of mass M and radius R. Pass the string around a pulley and attach it to a weight, whose mass is exactly half the mass of the hoop. Then release the system from rest. Describe the subsequent motions of the yo-yo and the weight. You may use equations to arrive at your answer, but you must state your result in plain sentences.

consider the forces involved. The static Puzzle involves the concept of rolling without slipping (because of the pulley) and can be solved completely with equations and subsequent English sentences of explanation. The dynamic Puzzle, however, requires some visual analysis and understanding of how the speed with which the mass falls is related to the physics quantities such as angular momentum and moment of inertia. It is clear from use of both static and Physlet-based questions that students who understand how to solve one of these sorts of questions do not necessarily know how to solve the other, so incorporating both types is an effective way to broaden and deepen all the students' understanding.

6 Conclusion

Based on our results, we believe that Physlets can be valuable tool for creating interactive curricular material designed around the needs of the student. We have investigated using Physlets to alter existing curricular material. However, the greatest potential of Physlets will probably come from using Physlets to ask questions in ways which can not be done on paper.

When developing a Physlet-based problem it is important to ascertain a clear instructional purpose. Using a Physlet cosmetically to merely enhance visualization of a question is gratuitous. For maximum benefit, Physlet questions should require students to interact with the simulation. Students should be required to

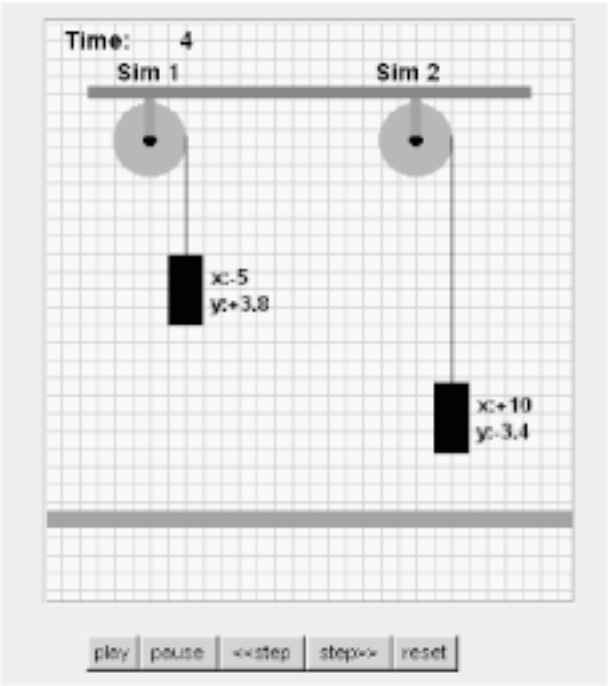


Fig. 6. Physlet-based JiTT Moment of Inertia Puzzle Question: Rank simulation 1 and simulation 2 from least to greatest in terms of the moment of inertia of the wheel, the tension in the string, and the total angular momentum about the wheel's axle after 4 seconds. The hanging weights have identical mass.

collect data, either numerically or visually. Research shows that if interaction is required, Physlets may influence how a student responds. Therefore, Physlet-based problems may be a more valid way to measure conceptual reasoning. The media-focus also makes them more challenging than traditional problems since novice solution strategies leave the student in despair.

Physlet problems are dynamic problems. Not only do they help students visualize a situation, they encourage the student to solve a problem the way a physicist solves a problem; that is, to consider the problem conceptually, to decide what method is required and what data to collect, and finally to analyze the data. It is akin to an open-ended laboratory experiment where students are not given instructions, but merely a question. They must decide what data to collect and how to most efficiently collect it. This quality seems to make Physlets well suited for evaluating students' application of conceptual understanding to numerical problems and helping students identify weaknesses in conceptual understanding.

7 Acknowledgements

Portions of the work presented here are based on published and unpublished work in collaboration with Aaron Titus and Evelyn Patterson. We would also like to thank Harvey Gould and Larry Cain for the many helpful comments in reviewing this manuscript.

The authors would like to acknowledge the National Science Foundation, grant DUE-9752365, for its support of Physlets.

References

1. *Physlets: Teaching Physics with Interactive Curriculum Material*, W. Christian and M. Belloni, Prentice Hall, Upper Saddle River, 2001; http://webphysics.davidson.edu/applets/applets.html.
2. R. R. Hake, *Interactive-engagement vs. traditional methods*, Am. J. Phys., **66**, 64 74 (1998);
   T. E. Sutherland and C. C. Bonwell, eds., *Using active learning in college classes; a range of options for faculty*, Jossey-Bass, San Francisco (1996).
3. D. R. Sokolo , *Using Interactive Lecture Demonstrations to Create an Active Learning Environment*, The Physics Teacher, 35, 340 (1997).
4. B. Thacker, *Comparing Problem Solving Performance of Physics Students in Inquiry-based and Traditional Introductory Courses*, American Journal of Physics, 62, 627-633. (1994).
5. *Just-in-Time Teaching: Blending Active Learning with Web Technology*, G. M. Novak, E. T. Patterson, A. D. Gavrin, and W. Christian, Prentice Hall, Upper Saddle River, 1999.
6. *Peer Instruction: A Users Manual*, E. Mazur, Prentice Hall, Upper Saddle River, 1997.
7. *Tutorial in Introductory Physics*, L. McDermott and P. S. Sha er, Prentice Hall, Upper Saddle River, 1998.
8. R. Beichner, *The Impact of Video Motion Analysis on Kinematics Graph Interpretation Skills*, American Journal of Physics, 64, 1272-1277. (1997).
9. A. Titus, *Integrating Video and Animation with Physics Problem Solving Exercises on the World Wide Web*, Ph.D. dissertation. North Carolina State University: Raleigh, NC. (1998).
10. M. Dancy, *Investigating Animations for Assessment with an Animated Version of the Force Concept Inventory*, Ph.D. dissertation. North Carolina State University: Raleigh, NC. (2001).
11. M. Dancy and B. Beichner.*Does Animation Influence the Validity of Assessment?*, (in preparation for the Journal of Research in Science Teaching).
12. D. Hestenes, M. Wells, and G. Swackhamer.*Force Concept Inventory*, The Physics Teacher, 30, 141 158 (1992).

Computation in Undergraduate Physics: The Lawrence Approach

David M. Cook

Department of Physics
Lawrence University
Appleton, WI 54912
USA
david.m.cook@lawrence.edu

Abstract. Most efforts using computers in physics curricula focus on introductory courses or individual upper-level courses. In contrast, for a dozen years the Lawrence Department of Physics has been striving to embed the use of general purpose graphical, symbolic, and numeric computational tools throughout our curriculum. Developed with support from the (US) National Science Foundation, the Keck Foundation, and Lawrence University, our approach involves introducing freshman to tools for data acquisition and analysis, offering sophomores a course that introduces them to symbolic, numerical, and visualization tools, incorporating computational approaches alongside traditional approaches to problems in many intermediate and upper level courses, and making computational resources available so that students come to see them as tools to be used routinely *on their own initiative* whenever their use seems appropriate. A text reflecting the developments at Lawrence is in preparation, will undergo beta testing in 2001-02, and will be published in January, 2003. Details about the Lawrence curricular approach and the emerging text can be found from links at www.lawrence.edu/dept/physics.

For a dozen years or more (and with support from the National Science Foundation[1], the W. M. Keck Foundation[2], and Lawrence University), we in the Department of Physics at Lawrence University[1] have been developing the computational dimensions of our upper-level curriculum[3,4,5,6]. We have built a computational laboratory that makes a wide spectrum of hardware and software available to students, developed an approach that introduces students to these resources and to prototypical applications, and drafted several hundred pages of instructional materials that, with support from a recent (US) National Science Foundation (NSF) grant[7], are currently being prepared for publication. This paper

1. lays out the underlying convictions that have guided the development of our approach to incorporating computation in an undergraduate curriculum,

[1] Lawrence University is a liberal arts college and conservatory of music with about 1250 students. The Department of Physics has five full time members and graduates an average of a dozen majors each year.

V.N. Alexandrov et al. (Eds.): ICCS 2001, LNCS 2073, pp. 1074–1083, 2001.
© Springer-Verlag Berlin Heidelberg 2001

2. describes the Lawrence curricular components,
3. discusses particularly the sophomore course that is the starting point for those students who pursue computation most aggressively, and
4. describes the instructional materials currently being prepared for publication.

1 Underlying Convictions

The primary tasks of those parts of an undergraduate physics program that focus on physics *majors* are to awaken in our students a full realization of the beauty, breadth, and power of the discipline and to help them develop both a secure understanding of fundamental concepts and the skills to use a variety of tools in applying those concepts. Among the tools, we at Lawrence would firmly include *computational* resources of several sorts. We believe

1. that our curricula must familiarize students
   a) with the functions and capabilities of at least one operating system.
   b) with the use of at least one good text editor (not word processor).
   c) with several types of computational tool, including
      i. a spreadsheet like Excel[8].
      ii. resources like IDL[9] and MATLAB[10] for numerical processing of numbers and arrays.
      iii. C and FORTRAN programming sufficient to permit comfortable use of subroutine packages like Numerical Recipes[11] and LSODE[12].
      iv. resources like MAPLE[13] and MATHEMATICA[14] for symbolic manipulation of expressions.
      v. resources like Kaleidagraph[15], IDL, MATLAB, and IRIS/NAG Explorer[16] for graphical visualization of complex data.
      vi. resources like LaTeX[17] and `tgif`[18] for preparing technical reports and manuscripts.
   d) with several types of symbolic and numerical analyses, including solving algebraic equations, solving ordinary and partial differential equations, evaluating integrals, finding roots, performing data analyses, fitting curves to experimental data, and manipulating images.
   e) with the assessment of accuracy in finite-precision arithmetic.
2. that students must be introduced early to these tools. An upper-level course in computational physics is a valuable curricular inclusion, but students need to become acquainted with computational resources long before they have either the mathematical or the physical background to profit from a rigorous computational physics course.
3. that use of computational resources must permeate the curriculum.
4. that the *initial* encounter with computational tools cannot be effectively accomplished as an appendix to tasks given higher priority. Certainly, numerous examples drawn from physical contexts must be used to motivate study of techniques and tools, but the focus must be on the features and capabilities of the tools.

Table 1. The typical program of a Lawrence physics major. Courses shown in bold type have explicit computer content.

	Term I	Term II	Term III
Year 1:	Social Science Elective	**Intro Classical Physics**	**Intro Modern Physics**
	Calculus I	Calculus II	Calculus III
	Freshman Studies	Freshman Studies	Free Elective
Year 2:	**Electronics**	**Mechanics**	**E and M**
	Linear Algebra/ODE	Humanities Elective	Humanities Elective
	Free Elective	Free Elective	Free Elective
Year 3:	**Quantum Mechanics**	**Advanced Laboratory**	Physics Elective
	Language	Language	Language
	Social Science Elective	Free Elective	Free Elective
Year 4:	Free Elective*	Physics Elective	Physics Elective
	Diversity Elective	Free Elective	Diversity Elective
	Free Elective	Free Elective	Free Elective

* Often independent research in physics.

In the broadest of terms, we should be structuring our curricula so that, ultimately, students will recognize when a computational approach may have merit and will be prepared to pursue that approach confidently, fluently, effectively, knowledgeably, and independently whenever they deem it appropriate.

The Lawrence approach to nurturing the abilities of students to use computational resources is active; it compels students to play a personal role in their own learning; it forces students to defend their work in writing; it gives students practice in preparing and delivering oral presentations; it encourages students to work in groups; it permeates our curriculum; and, more than any other objective, it develops the students' abilities to operate in this arena *on their own initiative*.

2 The Curricular Context

An efficient way to describe the Lawrence approach is to track the computational experience of an entering freshman physics major as she moves towards graduation four years later. Each year, full-time students at Lawrence take three courses in each of three ten-week terms. Class periods are 70 minutes long, and a one-term course translates officially into 3-1/3 semester hours. While there are many variations, the typical program of a student pursuing a physics major is shown in Table 1. This table also shows the area—though not necessarily the actual term—of courses needed to satisfy general education requirements. The minimum physics major is satisfied by ten courses in physics, seven of which are

Table 2. Available Electives. Again, courses shown in bold type have explicit computer content.

• Thermal Physics	• **Advanced E and M**
• Optics	• **Math Methods**
• Solid State Physics	• **Advanced Mechanics**
• Advanced Modern Physics	• Plasma Physics
• Laser Physics	• Tutorial in Physics

• **Computational Tools in Physics**
• Independent Study in Physics

explicitly stipulated, and four courses in mathematics. Courses shown in bold type direct students explicitly to the computer and, in most cases, include some instruction in one or more of our computational resources.

Available physics electives are shown in Table 2. Again, entries in bold type make explicit use of computational resources. In the other courses, students use those resources regularly on their own initiative. Majors are *required* to take three courses from the top group of nine and *may* take as many as five more from the entire spectrum before exceeding an institutionally imposed limit of 15 courses in any single department. Tutorials and independent studies, the latter being elected by nearly all senior majors and sometimes extending over more than one term and leading to honors in independent study at graduation, offer a vehicle for students to study topics not included in our regular course offerings.

3 The Computational Components

Prospective physics majors at Lawrence first encounter computational approaches in the introductory courses, whose laboratory is equipped with Macintosh computers, Vernier ULI cards[19], and a variety of sensors. Beyond LoggerPro, students have access both in the laboratory and elsewhere on campus to Excel and Kaleidagraph. Exercises assigned in the laboratory routinely involve automated data acquisition, statistical analysis, and curve fitting; exercises assigned in lectures occasionally send students to the laboratory computers for graphing results or pursuing numerical solutions to Newton's laws with editable Excel templates. By the end of the freshman year, prospective majors have already developed some skills in the use of computational tools, particularly skills of value in the laboratory.

Beyond the freshman year, students—of course—continue to use Excel and Kaleidagraph, but they also have access to our Computational Physics Laboratory (the CPL), which is equipped with six Silicon Graphics UNIX workstations, monochrome and color printers, and software in all the categories enumerated above. Each student has an account in this departmental facility, and each is entitled to a key both to the CPL and to the building, so each has 24/7 access to the CPL.

To help sophomores become confident, regular users of the CPL, we offer a course called *Computational Tools in Physics*, to be described in the next section. Even those sophomores who don't elect this course, however, encounter two short computational workshops—one on IDL and the other on MAPLE—in our *required* sophomore mechanics course. Thus, *all* sophomores have at least a small, *forced* exposure to the CPL, and some—but unfortunately not all—sophomores have a fully comprehensive introduction to the available capabilities. Subsequent theoretical and experimental courses alike offer students many opportunities to continue honing their computational skills and, depending on the instructor, some of these courses will direct students explicitly to the CPL for an occasional exercise. Most senior capstone projects will use the resources of the CPL, at least for visualization of data and/or preparation of reports. Some projects, notably those in fluid mechanics, musical acoustics, xray diffraction, mapping of astrophysical data, and multiphoton quantum transitions, have made extensive use of these facilities. Some physics students use the CPL in conjunction with courses in other departments, particularly mathematics.

4 The Sophomore Course

An elective course called *Computational Tools in Physics* is the starting point in our nurturing of our students' abilities to take full advantage of the resources of the CPL. Currently, this full-credit course is offered in three 1/3-credit segments, one in each of the three terms of our academic year. Its topics are coordinated with the required courses taken by sophomore majors. The first term focuses on acquainting students with the rudimentary capabilities of our CPL. It starts with a tutorial orientation to UNIX (1 week) and then addresses array processing and graphical visualization using IDL (2 weeks), publishing scientific manuscripts using LaTeX and `tgif` (1 week), graphical visualization using IRIS/NAG Explorer (2 weeks), symbolic manipulations using MAPLE (2 weeks), and circuit simulation using SPICE[20] (2 weeks). In each class, students are introduced to a particular computational tool. Then, each student works several exercises, ultimately turning in written solutions prepared with LaTeX. The half-dozen class sessions in the term provide only orientation and motivation; students are expected to exhibit a fair bit of personal independence and aggressiveness as they progress from the starting point provided by the classes to the knowledge and skill needed to finish the assignments.

The second term is coordinated with an intermediate course in classical mechanics, for which Barger and Olsson[21] is the current text, and focuses on symbolic and numerical approaches to ordinary differential equations (ODEs). In the first half of the term, the course covers symbolic solution of ODEs and Laplace Transforms with MAPLE (2 weeks), numerical solution of ODEs with IDL (2 weeks), and numerical solution of ODEs with FORTRAN programs and LSODE (1 week). Each student completes this term by carrying out an extended project that culminates in a written paper and a 20-minute oral presentation to

the class. Topics like the three-body problem, coupled oscillators, the compound pendulum, anharmonic oscillators, and chaos have been addressed.

The third term is coordinated with an intermediate course in electricity and magnetism, for which Griffiths[22] is the current text, and focuses on symbolic and numerical integration. In the first six weeks of the term, the course covers symbolic and numerical integration with MAPLE and IDL (2 weeks), numerical integration using FORTRAN programs and Numerical Recipes (2 weeks), and root finding using MAPLE, IDL, and Numerical Recipes (2 weeks). This term also concludes with an extended project, written paper, and 20-minute oral presentation. Topics like electric fields and potentials, magnetic fields, Fourier analysis, non-linear least squares fitting, and global positioning systems have been addressed.

5 The Emerging Text

The NSF grant[7] already mentioned provides support for converting the experience acquired and the library of instructional materials developed at Lawrence into a flexible publication[23] as a resource for other institutions. That we don't all use the same spectrum of hardware and software, however, poses a major challenge. The variety of options and combinations is so great that any single choice (or coordinated set of choices) is bound to limit the usefulness of the end result to a small subset of all potentially interested users. The strategy adopted to address that challenge involves assembling different incarnations of the basic materials from a wide assortment of components, some of which—the generic components—will be included in all incarnations and others of which—those specific to particular software packages—will be included only if the potential user requests them. Thus, the specific software and hardware treated in any particular incarnation will be microscopically "tailor-able" to the spectrum of resources available at the instructor's site. One incarnation, for example, could include the generic components and only the components that discuss IDL, MAPLE, C, and LaTeX while another might include the generic components and the components that focus on MATLAB, MATHEMATICA, and FORTRAN (including Numerical Recipes).

While the materials are still very much being refined, the present tentative table of contents includes the chapters and appendices listed in Table 3. In this structure,

1. Chapter 1 stands alone; chapters 2–5 introduce the general features of an array processor, a computer algebra system, a programming language, and the numerical recipes library; Chapters 6–12 address several important categories of computational processing; and the appendices introduce the use of an operating system, a publishing system, and a program for producing drawings.
2. In addition to the options explicitly indicated, later chapters also contain internal options that are not shown.

Table 3. Tentative table of contents for the book tentatively titled *Computation and Problem Solving in Undergraduate Physics*.

1. Overview
2. Introduction to IDL *and/or* MATLAB *and/or* ...
3. Introduction to MACSYMA *and/or* MAPLE *and/or Mathematica and/or* ...
4. Introduction to Programming in FORTRAN *and/or* C *and/or* ...
5. Introduction to Numerical Recipes
6. Solving Ordinary Differential Equations
7. Introduction to LSODE
8. Evaluating Integrals
9. Finding Roots
10. Solving Partial Differential Equations
11. Data Analysis/Curve Fitting
12. Fourier Analysis and Image Processing

A. Introduction to UNIX *and/or* Windows *and/or* ...
B. Introduction to LaTeX *and/or* Word *and/or* ...
C. Introduction to TGIF *and/or* ...

3. The order of presentation *in the book* does not compel any particular order of treatment *in a course or program of self-study*. While some later sections depend on some earlier sections, the linkages are not particularly tight.

While the objective is for students to become fluent in the use of a spectrum of computational tools—and the chapters are organized by program or by computational technique, the motivation throughout lies in and all examples are drawn from physical contexts.

Chapter 2, the tentative table of contents for the MATLAB version of which is shown in Table 4, represents chapters that introduce basic features of an application program, specifically a program for processing arrays of numbers and creating graphical visualizations of one-, two-, and three-dimensional data sets. The bulk of the chapters in this category are structured as tutorials and lean in some measure on vendor-supplied documentation and on-line help to encourage and guide self-study.

Shown in Table 5, the structure of Chapter 8 on evaluating integrals exemplifies that of all of the chapters on various computational techniques. Presumably, before approaching any particular section in this chapter, the student would have studied the relevant sections in earlier chapters. The first section, whose detail is laid out in the next paragraph, sets several physical problems, the successful addressing of which benefits from exploitation of a computational tool. The second section describes how one might use a symbolic tool in application to some of the problems set in the first section. Save for the last, the remaining sections describe suitable numerical algorithms generically and then illustrate how those algorithms can be invoked in a variety of ways. The final section lays out several exercises that students can use to hone their skills. Sections 8.1, 8.3, and 8.7 would be included in all versions of the chapter; each individual instructor

Table 4. Sections in the chapter on MATLAB. The IDL chapter is similar.

2.1	Beginning a Session with MATLAB
2.2	Basic Entities in MATLAB
2.3	A Sampling of MATLAB Capabilities
2.4	Properties, Objects, and Handles
2.5	Saving/Retrieving a MATLAB Session
2.6	Loops/Logical Expressions/Conditionals
2.7	Reading Data from a File
2.8	On-line Help
2.9	Command Files
2.10	Eigenvalues and Eigenvectors
2.11	Graphing Functions of One Variable
2.12	Making Hard Copy
2.13	Graphing Functions of Two Variables
2.14	Graphing Functions of Three Variables
2.15	Graphing Vector Fields
2.16	Animation
2.17	Advanced Graphing Features
2.18	Miscellaneous Occasionally Useful Tidbits
2.19	References
2.20	Exercises

would select from the indicated options only those that are appropriate to that instructor's site.

Yet one level further down in the envisioned structure, Table 6 shows the present list of sample problems for Chapter 8. They range over several subareas of physics and reveal that evaluation of integrals, perhaps as functions of one or more parameters, plays an important role in many areas of physics.

Even among sites that use the same spectrum of hardware and software, however, some aspects of local environments are still unique to individual sites. Rules of citizenship, practices and policies regarding accounts and passwords, the features and elementary resources of the operating system, the structuring of public directories, backup schedules, after-hours access, licensing restrictions in force on proprietary software, and numerous other aspects are subject to considerable local variation. The emerging book will make no attempt to constrain local options in these matters. Throughout the book, individual users are directed to a publication called the *Local Guide* for site-specific particulars. A suggested template for that guide will be provided, but it will require editing to reflect local practices.

The desired flexibility to tailor the book to a variety of circumstances would be unattainable without LaTeX. Of particular significance, LaTeX is able to decide in response to conditional statements controlled by Boolean flags which files should be included and which omitted in any particular processing run. The procedure has already passed its proof-of-concept trial, and the publisher

Table 5. Sections in the chapter on integration.

8.1	Sample Problems
8.2	Evaluating Integrals Symbolically with MACSYMA *and/or* MAPLE *and/or Mathematica and/or* ...
8.3	Algorithms for Numerical Integration
8.4	Evaluating Integrals Numerically with IDL *and/or* MATLAB *and/or* ...
8.5	Evaluating Integrals Numerically with MACSYMA *and/or* MAPLE *and/or Mathematica and/or* ...
8.6	Evaluating Integrals Numerically with FORTRAN *and/or* C *and/or* Numerical Recipes
8.7	Exercises

Table 6. Sample problems in Section 8.1.

8.1.1.	One-Dimensional Trajectories
8.1.2.	Center of Mass
8.1.3.	Moment of Inertia
8.1.4.	Large-Amplitude Pendulum (Elliptic Integrals)
8.1.5.	The Error Function
8.1.6.	The Cornu Spiral
8.1.7.	Electric/Magnetic Fields and Potentials
8.1.8.	Quantum Probabilities

Brooks-Cole is committed to refining this essential procedure so as to be able to make a commercially feasible product. Further, once the structure has been fully worked out, contributiions from other authors may be added, so—over time—the product could expand to accommodate a wider and wider spectrum of hardware and software and include topics not originally incorporated. We sincerely hope that this book, emerging as it has from a dozen years of experience and development at Lawrence, will support efforts at other institutions to embed meaningful computational components in their undergraduate curricula.

References

1. NSF ILI Grant DUE-8851685 for $49433, awarded in June, 1988, for a project entitled "Scientific Workstations in Undergraduate Physics"; NSF ILI Grant DUE-9350667 for $48777, awarded in June, 1993, for a project entitled "Partial Differential Equations in Advanced Undergraduate Physics."
2. Keck Grant #880969 for $200,000, awarded in June, 1988, to support integrating scientific workstations into the physics curriculum; Keck Grant #931348 for $250,000, awarded in December, 1993, to support the enhancement of advanced theoretical, computational and experimental courses.

3. David M. Cook, "Computers in the Lawrence Physics Curriculum" Part I, Comput. Phys. **11**(3; May/Jun, 1997), 240–245; Part II, Comput. Phys. **11**(4; Jul/Aug, 1997), 331–335.
4. David M. Cook, "Incorporating Uses of Computational Tools in the Undergraduate Physics Curriculum" in *Computing in Advanced Undergraduate Physics*, the proceedings of a Sloan-supported conference held at Lawrence University, 13–14 July 1990, edited by David M. Cook and published in November, 1990, by Lawrence University.
5. David M. Cook, "Computational Exercises for the Upper-Division Undergraduate Physics Curriculum," Comput. Phys. **4**(3; May/June), 308–313 (1990).
6. David M. Cook, "Introducing Computational Tools in the Upper-Division Undergraduate Physics Curriculum," Comput. Phys. **4**(2; Mar/Apr), 197–201 (1990)
7. NSF CCLI-EMD Grant DUE-9952285 for $177,000, awarded in February, 2000, for a project entitled "Strengthening Computation in Upper-Level Undergraduate Physics Programs."
8. Available from Microsoft, Seattle, Washington.
9. Available from Research Systems, Inc., Boulder, Colorado.
10. Available from The MathWorks, Inc., Natick, Massachusetts.
11. Available from Numerical Recipes Software, Cambridge, Massachusetts.
12. LSODE (the Livermore Solver for ODEs) is a component in the package ODEPACK, which is in the public domain and is available for ftp transfer from appropriate sites, e.g., `www.netlib.org`.
13. Available from Waterloo Software, Waterloo, Ontario.
14. Available from Wolfram Research, Inc., Champaign, Illinois.
15. Available from Synergy Software, Reading, Pennsylvania.
16. Available from Numerical Algorithms Group, Downers Grove, Illinois.
17. Available for many platforms via ftp transfer from several archives around the world. For information, start at `www.tug.org`.
18. Tgif is a program for drawing assorted diagrams and writing those descriptions in files of several different types. For information, go to `www.ucla.edu` and search for tgif.
19. Available from Vernier Software and Technology, Beaverton, Oregon.
20. SPICE is a tool for simulating the behavior of electric circuits. For information, start at `www.berkeley.edu` and search for SPICE.
21. Vernon D. Barger and Martin G. Olsson, *Classical Mechanics: A Modern Perspective* (McGraw-Hill, New York, 1995), Second Edition.
22. David J. Griffiths, *Introduction to Electrodynamics* (Prentice-Hall, Upper Saddle River, New Jersey, 1999), Third Edition.
23. David M. Cook, *Computation and Problem Solving in Undergraduate Physics* (Brooks-Cole, Pacific-Grove, CA, expected January, 2003).

Recent Developments of a Coupled CFD/CSD Methodology

Joseph D. Baum[1], Hong Luo[1], Eric L. Mestreau[1], Dmitri Sharov[1], Rainald Löhner[2], Daniele Pelessone[3], and Charles Charman[4]

[1] Center for Applied Computational Sciences, SAIC, McLean, VA 22102, USA
{baum, luo, mestreau, sharov}@apo.saic.com
[2] CSI, George Mason University, Fairfax, VA 22030, USA
lohner@rossini.gmu.edu
[3] Engineering and Software System Solutions, Solana Beach, CA 92075, USA
[4] General Atomics, San Diego, CA 92121,USA

Abstract. A recently developed loose-coupling algorithm that combines state-of-the-art Computational Fluid Dynamics (CFD) and Computational Structural Dynamics (CSD) methodologies has been applied to the simulations of weapon-structure interactions. The coupled methodology enables cost-effective simulation of fluid-structure interactions with a particular emphasis on detonation and shock interaction. The coupling incorporates two codes representing the state-of-the-art in their respective areas: FEFLO98 for the Computational Fluid Dynamics and DYNA3D for the Computational Structural Dynamics simulation. An application of the methodology to a case of weapon detonation and fragmentation is presented, as well as fragment and airblast interaction with a steel wall. Finally, we present results of simulating airblast interaction with a reinforced concrete wall, in which concrete and steel rebar failure and concrete break-up to thousands of chunks and dust particles are demonstrated.

1 Introduction

Several classes of important engineering problems require the concurrent application of CFD and CSD techniques. Among these are: a) Shock/structure interactions; b) Aeroelasticity of flexible thin flight structures; c) Hypersonic flight vehicles (thermal-induced deformations); d) Deformation of highly flexible fabrics; and e) Vehicles with variable geometry. Currently, these problems are solved either iteratively, requiring several cycles of "CFD run followed by CSD run", or by assuming that the CFD and CSD solutions can be decoupled.

The various efforts to develop a fluid/structure coupling can be classified according to the complexity level of the approximations used for each of the

V.N. Alexandrov et al. (Eds.): ICCS 2001, LNCS 2073, pp. 1087–1097, 2001.
© Springer-Verlag Berlin Heidelberg 2001

domains. Approximations of the Partial Differential Equations for the structural mechanics range from simple 6 DOF integration to finite elements with complex models for elasto-plastic materials with rupture laws and contact. Similarly, the fluid dynamics approximations of the PDEs range from the potential flow (irotational, inviscid, isentropic flows) to the full Navier-Stokes set of equations. Our present research interests focus on non-linear applications, in particular, structures that experience severe deformations due to blast loads, aerodynamic or aero-thermodynamic loads. The fluid approximation chosen is either Euler or Reynolds-Averaged Navier-Stokes. On the structure side, elasto-plastic materials with rupture criteria are used.

In this study, the coupled CFD/CSD methodology is applied to the simulation of weapon detonation and fragmentation. This application constitutes a very severe test to the numerical methodology as it requires modeling of several complex, interacting physical phenomena: a) Detonation wave initiation and propagation; b) CSD modeling of case expansion and fragmentation; c) The transfer of rigid fragments from the CSD to the CFD modules; d) Blast wave expansion through the breaking case, diffracting about the flying fragments; e) Flight of thousands of rigid bodies, each treated as a separate, free-flying body, where its trajectory and velocity are determined by balance of forces and moments; and f) Fragments and airblast impact on the structure and the resulting structural deformation.

Two approaches can be used to tackle fluid/structure interaction. The so-called 'tight coupling' approach requires solving both CFD & CSD as one coupled set of equations, and would require the complete rewrite of both solvers. The second approach, termed 'loose coupling', decouples the CFD and CSD sets of equations and uses projection methods to transfer interface information between the CFD and CSD domains. We adopted the latter method. By building on pre-existing and well- established codes, a loose-coupled solver can be assembled with minimum modifications to either of the two solvers. The modularity is kept by the addition of a 'controller' code, which handles the transfer of information between the different solvers [7], [11], [3]. This code handles non-matching meshes at the interface and incorporates conservative interpolation schemes and fast techniques for neighbor search. It deduces automatically the correspondence between fluid and structure points without any user input. Time synchronization between the CFD and CSD solvers is also managed by the controller code, which uses a leap-frog approach.

1.1 The Current Numerical Methodology

Mesh generation was performed using FRGEN3D, an advancing front based grid generator [9]. This mesh generator is also included in the flow solver, FEFLO98, to handle mesh regeneration on the fly. The mesher requires the input of CAD surfaces and lines. Very complex shapes can now be meshed in a matter of hours once the model is properly defined [1], [2], [3]. However, assembling the CAD definition of the model still remains the bottleneck, consuming large amounts of man-hours. To remedy this deficiency, we have developed a dedicated graphic

pre-processor, to promptly handle the specifics of the mesher/solver such as: boundary conditions, element size definition and automatic generation of the structural model from the predefined fluid domain. The pre-processor also provides extensive data checking allowing a considerable gain in productivity. For the current study, both CFD & CSD meshes were generated with FRGEN3D. The CFD mesh is composed of tetrahedral elements in the volume and triangles on the surfaces. The CSD mesh includes beams, quad and triangle shells (quads corresponds to the concatenation of 2 triangles) and bricks for the volume. The bricks result from the cut of tetrahedral elements. Although the angles of a typical hex are less than perfect, extensive testing against perfect-angle bricks for both linear and nonlinear tests, produced almost identical results. This, nevertheless, necessitated the replacement of the Belytschko-Tsay hourglass control model (default model in DYNA3D [15]), with the Flanagan-Belytschko hourglass control model (model no. 3 in DYNA3D [5]), incurring a 30% performance penalty.

The flow solver is FEFLO98, a 3-D adaptive, unstructured, edge-based hydrosolver based on the Finite-Element Method Flux-Corrected Transport (FEM-FCT) concept [8]. It solves the Arbitrary Lagrangean-Eulerian (ALE) formulation of the Euler and Reynolds-Averaged turbulent, Navier-Stokes equations. The high order scheme used is the consistent-mass Taylor-Galerkin algorithm. Combined with a modified second-order Lapidus artificial viscosity scheme, the resulting scheme is second-order accurate in space, and fourth-order accurate in phase. The spatial adaptation is based on local H-rfinement, where the refinement/deletion criterion is a modified H2-seminorm [10] based on a user- defined unknown. For detonations and shock wave diffraction simulations, the critical parameter used for the refinement/deletion criteria is density. The explosive detonation is modeled using a JWL equation of state with afterburning. To enhance computational efficiency, the portion of the fluid domain not reached by the blast wave is automatically deactivated.

The structural dynamics solver is DYNA3D [15], an unstructured, explicit finite element code. DYNA3D is well suited for modeling large deformations and provides a good base for non-linear materials with elasto-plastic compartmental laws with rupture. DYNA3D incorporates a large library of materials and various equations-of-state, as well as many kinematic options, such as slidelines and contacts. Furthermore, DYNA3D is a well proven and benchmarked solver used extensively in the CSD community.

2 Numerical Results

2.1 Weapon Fragmentation Study

The coupled technology has been applied to the simulation of the detonation and fragmentation of an experimental weapon. The bomb hangs tip-down at the center in a reinforced concrete room The thick- walled steel weapon is top (i.e., base) ignited. The detonation front propagates from the base to the tip at the C-J detonation velocity, as prescribed in the detonation model (essentially, the

program burn model of DYNA3D). Initially, the CFD domain consisted of two separate regions: the domain inside the case is modeled using the JWL EOS, while the ambient atmosphere outside is modeled using a perfect gas EOS. Once fragmentation occurred, the two topologies merged and the complete domain is modeled using the JWL EOS.

The structural response (case expansion) is modeled using GA-DYNA [12], [13], the General Atomics version of DYNA3D. Several CSD meshes of this weapon were tested, using either 8-node hexahedral elements or brick-like parallelepipedal elements, and varying the number of elements from 748 to 8228. The results presented here were obtained with 748 brick elements with a single element across the thickness of the casing. The fragment size distribution for the present simulation **is prescribed**. This value was obtained by averaging fragment sizes from several arena tests. A more accurate procedure is described below [14].

After ignition, as the detonation wave propagates from the base to the tip, the high-pressure detonation products force the case to expand. The structural elements fail once the element strain (averaged over all faces) exceeds 100%. The strain criterion for failure is computed at the center of each element. Each failing fragments is then treated as a separate rigid body, for which the trajectory is computed using a 6 DOF integrator linked to the contact algorithms. Once bricks fail, fluid elements are introduced into the narrow gaps separating the fragments. The gaps are of the order of a millimeter, which would result in unacceptable small fluid elements and small integration time step. The gap size was increased by shrinking the fragments uniformly around the center of gravity. The topology change due to the breakup requires remeshing of at least part of the domain, a CPU intensive process that is allowed to occur only every 5-8 μs. Thus, the CSD code maintains a list of failed elements, and shrinks them only when allowed.

One important aspect of this class of simulations is the large size disparity between the critical length scales. After fragment break-up, the gap between fragments is several **millimeters**. In contrast, the average fragment length is about ten **centimeters**, and the room length is of the order of ten **meters**. The large disparity in dimensions forced us to attach to each flying fragment entities called grid sources. The sources enforce local pre-specified element size, ensuring a uniform, high-resolution mesh about each fragment, and thereby reduce the number of local and global remeshings. Figures 1a through 1c show the CSD surface mesh, the CFD mesh on a planar cut through the weapon (not a plane of symmetry due to the lack of symmetry for this weapon), and the CFD surface mesh, respectively, at 550 μs. The results show the finely-resolved mesh within the initial HE zone, resulting from the application of a grid source placed along the center of the weapon. On the weapon we form a much finer CFD mesh than the CSD mesh, with rapid increase of mesh size with distance from the weapon. The range-of-influence of the centerline-placed source was specified to produce a fine-resolution mesh not just within the HE zone, but also around the complete volume in which fragmentation occurs after case expansion. As the fragments exit the fine-resolution zone imposed by the central source, CFD mesh resolution

is reduced to the level specified by the sources attached to the fragment. This can clearly be seen on the third row from the top, where mesh resolution is reduced from several elements across each fragment at t=0, to three elements at t=350 μs, and finally two elements at t=550 μs (Fig 1c). Finally, examination of the planar cut results (Fig 1b) shows that the mesh size dictated by the sources is dependent on the fragment and face size. Sources attached to the smaller moving faces yield a finer mesh resolution than those attached to the larger ones. Thus, at t=550 μs, the large fragments that are outside the core central source (such as the rows 4, 15 and 17), show coarse mesh resolution on the large faces, but fine resolution within the gaps and the near-by faces.

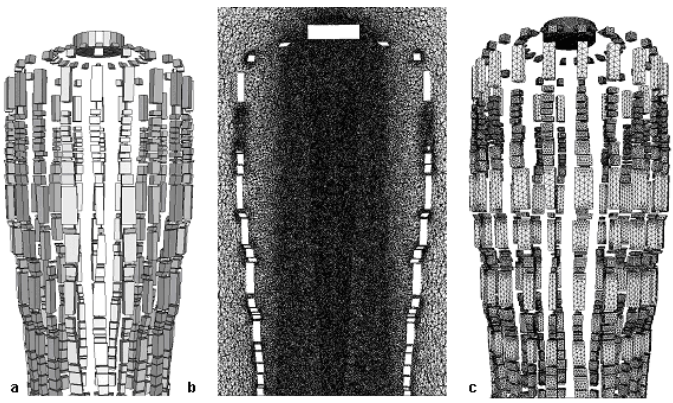


Fig. 1. CSD mesh, CFD mesh on a plane cut and CFD mesh on the surface of the weapon at t=0.550ms

During case expansion, the internal mesh velocity significantly exceeds the external velocity, resulting in case thinning. On average, while the CG of the element experiences a 100% strain (break-up criteria), the internal face expands about 145-160%, compared to about 70-80% for the external face. This indicates that during a significant portion of the expansion period, the internal face velocity is about twice the external face velocity.

Figures 2a through 2d show a sequence of snapshots. At each time, the panel shows the pressure and CFD mesh velocity contours on a planar cut through the weapon, and the CSD fragment velocity contours. Figure 2a at 131 μs shows detonation wave propagation down (from base to tip) as a planar front, and the radial expansion of the case. The first fragment break-up occurs at 94 μs, for the upper row attached to the heavy base. While the base itself does not fragment, as it does not expand significantly, the row of elements below fails due to shear, not tension. Similarly, the layer of fragments above the nose cone fails due to shear (Fig 2c at 370 μs). Detonation was completed at about t=263 μs and the shock reflected upward (Fig 2b).

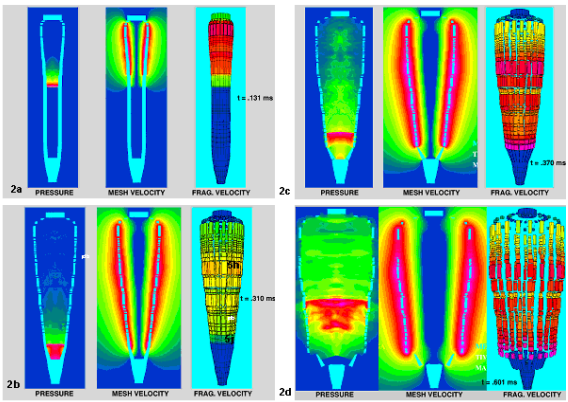


Fig. 2. Propagation of detonation wave and case fragmentation. Results show pressure, mesh velocity, and fragment velocity at times 131 μs, 370 μs, and 600 μs, respectively.

The relatively small spacing between the expanding fragments ensures that the high-pressure detonation products would be fairly contained within the expanding fragments for an extended period. The results demonstrate that even at t=0.6 ms (Fig 2d), the pressure within the core is significantly higher than outside.

The fragments achieve their terminal velocity within about 120-150 μs after detonation front passage. This is significantly slower then the acceleration period of 60-90 μs for a serrated weapon [4]. The final mass-averaged fragment velocity obtained for this simulation, using a strain break-up value of 1.0, was 787.6 m/sec. The experimental measured value was 752 m/sec. To examine the role of the break-up strain value on the final velocity we conducted three more simulations, at break-up strain values of 0.1 (V_f=675.53 m/sec), 0.5 (V_f=741.8 m/sec), and infinity (no break-up, V_f=859.7 m/sec). The results show that the experimental data indicates a break-up strain value of about 0.6, a value that corresponds to an internal expansion of about 90%, and external expansion of about 50%.

A total of five simulations were conducted under this study, investigating the role of the break-up strain value on the final fragmentation velocity distribution, and single vs. multiple elements through the case thickness. The initial mesh for each simulation included about 8.7 million elements, the final one about 19.2 million elements. The five simulations conducted averaged about five days on a SGI Origin 2000, using twelve to sixteen processors. Once remeshing parallelization was completed, CPU time was cut to less than a day.

2.2 Blast Impact on a Reinforced Concrete Wall

As the next step in the CFD/CSD coupling development effort, we applied the coupled methodology to the simulation of airblast interaction with reinforced

concrete wall. The model included two rooms, but only the connecting wall was modeled with the CSD code. All other structural components (e.g., other walls, floor and ceiling) were treated as rigid. The CFD domain consisted of 9,387 boundary points, 55,281 points and 296,751 elements. The wall included 81,101 nodes, 69,048 solid hexahedron elements in the concrete and 598 beam points in the steel rebars. While the CFD solution was non- adapted, three levels of mesh adaptation [12] were in the CSD model. The standard DYNA3D element erosion model was used to eliminate failed CSD elements.

Several new schemes were employed here. These include: 1) A recently developed crack propagation model [14] that takes advantage of the CSD H-refinement scheme. As the crack propagates through the material, mesh adaptation is used to ensure the accuracy of the stress wave propagation, and the accurate agglomeration of the elements into discrete fragments. This approach alleviates the need for expensive arena test data. The new model was validated against data for two test [14]; 2) The adaptation procedure ensures that each fragment contains several elements. As the elements fail and fragments are formed, each is treated by GA-DYNA as an independent body, with the appropriate volume, mass, momentum and energy. GA-DYNA then keeps track of fragment-to-fragment and fragment-to-wall interactions through a contact algorithm. GA-DYNA transfers the information to FEFLO98, which treats every fragment as a sphere, allowing for accurate momentum and energy exchange (e.g., drag and heat transfer); and 3) A new model that allows rebar data to be interpolated from enclosing elements, in contrast to the original DYNA3D that required all nodes to be on the rebar itself.

Figure 3 shows several snapshots taken during the simulation. Fig. 3a shows the CSD mesh as shown on the surface. Notice that the CSD elements were generated by splitting the CFD elements (as clearly seen on the sides). The steel rebars are shown in Fig 3b. The concrete material used was intentionally 'softened' to produce faster wall break-up (for testing and debugging purposes). Hence, the significant damage shown in Fig. 3c, after only 400 time steps. Each element face is given a uniform color corresponding to the value of the element damage parameter. No nodal averaging was performed. Figures 3d, 3e and Figs 3d, 3g show a pair of snapshots (front and back) taken early and late in the run, respectively. The figures show the computed geometry as realized by the CSD code (Figs 3d and 3f), and as realized by the CFD code (Figs 3e and 3g). While the CSD code integrates all structural matter, including all the produced debris, small particles and dust, not all information is transferred to the CFD code. Only information about large chunks and large fragments is transferred to the CFD code. These are treated by the CFD code as moving bodies. Thus, the CFD code computes the motion of hundreds of moving bodies, evaluating the forces acting on them and the resulting trajectories. Small particles and dust trajectories are carried by the CSD code, and only the momentum and energy transfer information is exchanged with the CFD code, so that the CFD code can accurately compute energy dissipation due the drag and thermal losses imposed by the flying smaller, cooler particles. Figures 3h and 3i show a superposition of

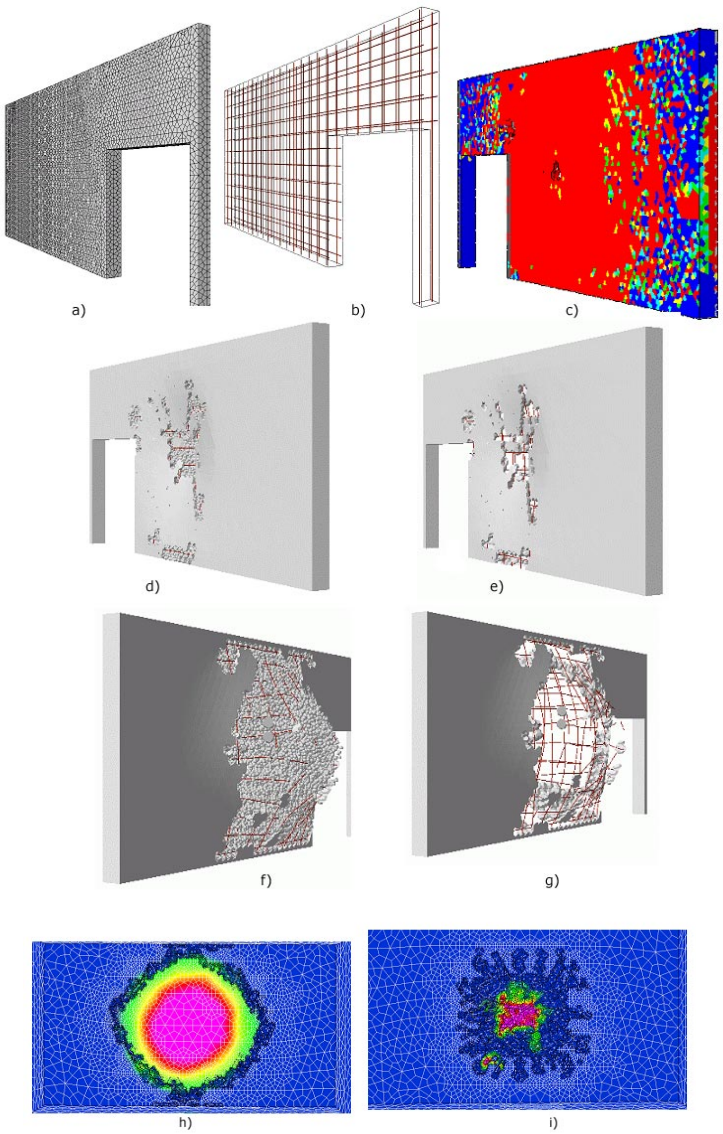


Fig. 3. This figure shows the initial CSD mesh, and the structural response to blast. The figures show the CSD surface mesh (Fig 3a); the rebar pattern (Fig 3b); the element damage parameter after 400 steps (Fig 3c); the CSD and CFD surface realizations at early time (Figs 3d and 3e), and late time (Figs 3f and 3g), respectively; a superimposed pressure contours and adapted CSD mesh at an early time on the front and back faces of the damaged wall (Figs 3h and 3i, respectively).

pressure contours and CSD mesh on both sides of the wall. The results show the typical damaged concrete pattern: a crown in the blast room and spallation web on the opposite side. Notice the complex connectivity through the concrete that allows the high pressure to emerge through the other side of the wall: from the peripheral crown to the centered spall zone. Three levels of mesh adaptation are shown in these figures. The adapted CSD mesh enables accurate prediction of the spallation, crack propagation, element failure and fragment formation, which expose the rebars on the spalled side.

2.3 Blast Fragment Impact on a Steel Chamber

The coupled CFD/CSD methodology was applied to the simulation of airblast and fragment interaction with a steel walled chamber. While the CFD solution was non-adapted, three levels of mesh adaptation [Pe97] were in the CSD model. The standard DYNA3D element erosion model was used to eliminate failed CSD elements.

The numerical predictions show that the impacting weapon fragments arrive ahead of the airblast (Fig 4a), punching holes through the plate (Figs 4b, 4c and 4e). Next, the pressure blast from the detonation tears the weakened plate apart (Figs 4d and 4f). The eroded plate elements were converted into particles that can interact with the rest of the structure. Contact conditions were enforced between all entities of the model, thus avoiding simulation break due to fragments coming in contact with each other and eliminating the CFD mesh in between.

Significant CPU cost reduction was achieved by the allowing the CSD code to model convection of the small broken pieces. The CFD code handles these pieces as spheres with the correct effective radius, modeling only the momentum (drag) and energy exchange between the blast and the spheres. Hence, the information transferred from the CSD to the CSD module is reduced to the minimal sphere data (radius, density, velocity vector and temperature).

3 Summary and Conclusions

A recently developed loose-coupling algorithm that combines state-of-the-art Computational Fluid Dynamics (CFD) and Computational Structural Dynamics (CSD) methodologies, has been applied to the simulation of weapon detonation and fragmentation. This application required modeling several complex and interacting physical phenomena. In addition to the loose coupling of two state-of-the-art codes, FEFLO98 and DYNA3D, several new routines were developed to allow better communications between the codes, especially during case fragmentation. The results demonstrate the ability of the coupled methodology to handle these processes and yield results that are in good agreement with experimental data. While other techniques may be used to model weapon fragmentation, the advantage of the coupled CFD/CSD methodology is that in addition to the fragment size and velocity distribution, it also yields an accurate description of the airblast environment. The resulting fragment and airblast predictions can then be used to predict the target response to the attack.

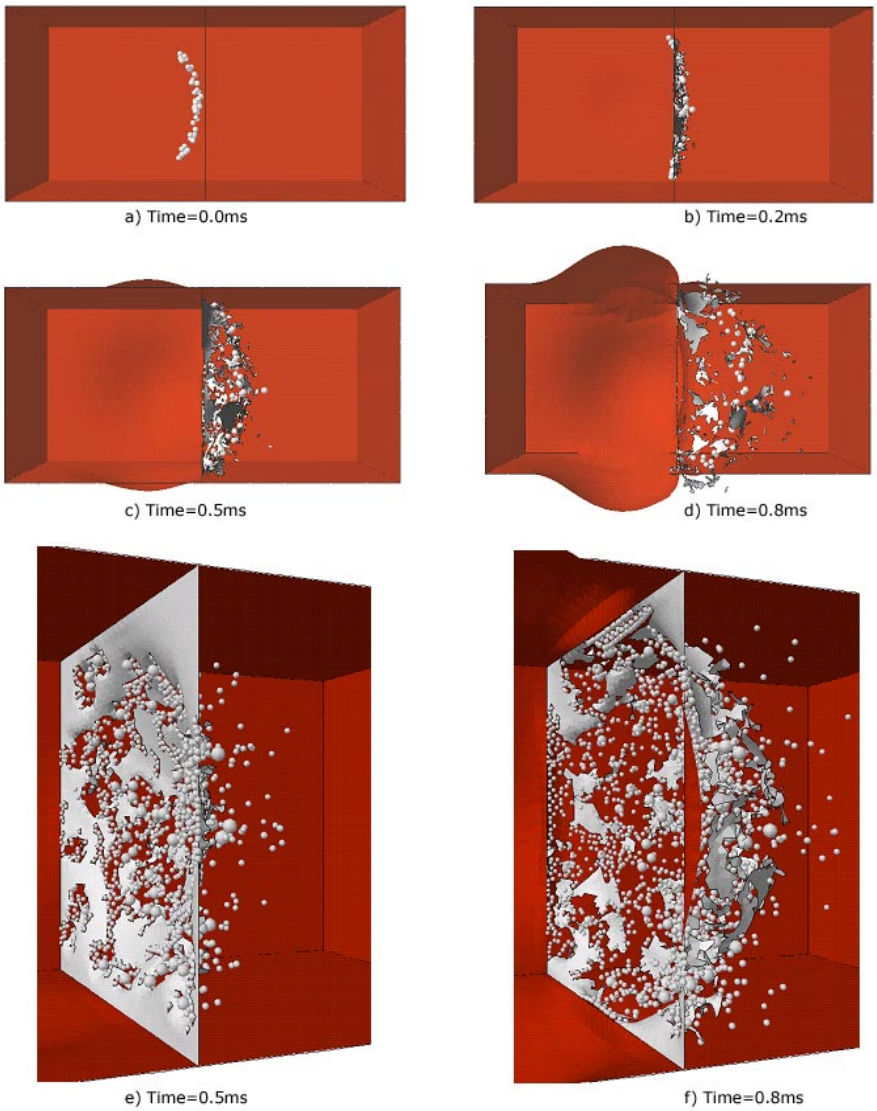


Fig. 4. Figures 4a shows the initial fragment position; Fig. 4b shows the surface immediately after fragment impact (t=0.2 ms); Figs 4c and 4e show the steel surface at t=0.5ms, after fragment impact but before airblast impact, while Figs 4d and 4f show the complete repture of the steel plate after blast impact.

4 Acknowledgements

This research effort was supported by the Defense Threat Reduction Agency. Dr. Michael E. Giltrud, served as the contract technical monitors. Computer time was generously provided by the DOD High Performance Computing Modernization Office (HPCMO).

References

1. J.D. Baum, H. Luo, and R. Löhner : Numerical Simulation of a Blast Inside a Boeing 747; AIAA-93-3091 (1993).
2. J.D. Baum, H. Luo and R. Löhner : Numerical Simulation of Blast in the World Trade Center; AIAA-95-0085 (1995).
3. J.D. Baum, H. Luo, R. Löhner, C. Yang, D. Pelessone and C. Charman : A Coupled Fluid/Structure Modeling of Shock Interaction with a Truck; AIAA-96-0795 (1996).
4. J.D. Baum, H. Luo and R. Löhner : The Numerical Simulation of Strongly Unsteady Flows With Hundreds of Moving Bodies; AIAA-98-0788 (1998).
5. T. Belytschko, and J.I. Lin : A Three-Dimensional Impact-Penetration Algorithm with Erosion; Computers and Structures, Vol. 25 No. 1, p 95, 1986.
6. D.J., Benson, and J.O. Hallquist : A single surface contact algorithm for the post-buckling analysis of shell structures; Computational Methods in Applied Mechanics and Engineering, Vol. 78, No. 2 p 141, 1990.
7. J.R. Cebral and R. Löhner : Conservative Load Transfer for Fluid-Structure-Thermal Simulations; Proc. 4th WCCM, Buenos Aires, Argentina, July (1998).
8. R. Löhner, K. Morgan, J. Peraire and M. Vahdati : Finite Element Flux-Corrected Transport (FEM-FCT) for the Euler and Navier-Stokes Equations; Int. J. Num. Meth. Fluids 7, 1093-1109 (1987).
9. R. Löhner and P. Parikh : Three-Dimensional Grid Generation by the Advancing Front Method; Int. J. Num. Meth. Fluids 8. 1135-1149(1988).
10. R. Löhner and J.D. Baum : Adaptive H-Refinement on 3-D Unstructured Grids for Transient Problems; Int. J. Num. Meth. Fluids 14, 1407-1419 (1992).
11. R. Löhner, C. Yang, J. Cebral, J.D. Baum, H. Luo, D. Pelessone and C. Charman : Fluid- Structure Interaction Using a Loose Coupling Algorithm and Adaptive Unstructured Grids; AIAA-95-2259 (1995).
12. D. Pelessone, and C.M. Charman : An Adaptive Finite Element Procedure for Structural Analysis of Solids; 1997 ASME Pressure Vessels and Piping Conference, Orlando, Florida, July (1997).
13. D. Pelessone and C.M. Charman : A General Formulation of a Contact Algorithm with Node/Face and Edge/Edge Contacts; 1998 ASME Pressure Vessels and Piping Conference, San Diego, Ca, July (1998).
14. D. Pelessone, C.M. Charman, R. Löhner and J.D. Baum : A new Crack Propagation Algorithm for modeling Weapon Fragmentation; in preparation.
15. R.G. Whirley and J.O. Hallquist : DYNA3D, A Nonlinear Explicit, Three-Dimensional Finite Element Code for Solid and Structural Mechanics - User Manual; UCRL-MA-107254 (1991), also Comp. Meth. Appl. Mech. Eng. 33, 725-757 (1982).

Towards a Coupled Environmental Prediction System

Julie L McClean[1], Wieslaw Maslowski[1], and Mathew Maltrud[2]

[1]Department of Oceanography, Naval Postgraduate School, Monterey, California, USA
mcclean@nps.navy.mil, maslowsk@nps.navy.mil
[2]Los Alamos National Laboratory, Los Alamos, New Mexico, USA
maltrud@lanl.gov

Abstract. Towards the realization of a global coupled air/ocean/ice predictive system for Navy needs, two high resolution modeling efforts are underway whose goals are the development and upgrading of the ocean and sea ice components. A 0.1°, 40-level global configuration of the Los Alamos National Laboratory (LANL) Parallel Ocean Program (POP) integration is being performed on an IBM SP3; this is the first time an ocean simulation of this size has been carried out. The Polar Ice Prediction System (PIPS) 3.0 uses a 1/12°, 45-level grid and covers all the northern ice-covered regions. The latter model and a 0.1°, 40-level North Atlantic only POP integration are compared with coarser resolution runs and observations, demonstrating the importance of high resolution to the representation of ocean circulation. Mean volume and heat transports into the Arctic are realistically simulated by PIPS 3.0.

1 Introduction

State-of-the-art super-computer technologies are providing the US Navy with the means to progress towards the realization of their vision of a high resolution operational global air/ocean/ice system for the prediction of environmental conditions. An understanding of the atmosphere-ice-ocean states and their variability in the form of short-term predictions of weather, ocean, and ice conditions are important to daily Naval operations and critical in the battlespace environment. Part of the modernization effort in the Navy involves the development of improved codes for ocean and sea ice simulations, which employ the best representations of sub-grid scale physical processes while taking advantage of available computer resources.

Two parallel efforts are underway that are making progress towards these goals by developing and upgrading the ocean and sea ice components of this future coupled Navy environmental prediction system. The spin-up of a global 1/10°, 40-level ocean model is underway; upon completion it will be delivered to the Fleet Numerical Me-

V.N. Alexandrov et al. (Eds.): ICCS 2001, LNCS 2073, pp. 1098–1107, 2001.
© Springer-Verlag Berlin Heidelberg 2001

teorological and Oceanographic Center (FNMOC) for testing and transition to the operational environment. The Polar Ice Prediction System (PIPS) 3.0, which will replace PIPS 2.0 providing forecasts of ice conditions in the Northern Hemisphere, uses a 1/12°, 45-level grid and covers the northern ice-covered oceans. It is currently being transitioned for operational use.

The ocean model being used in both the global and PIPS prediction systems is the Los Alamos National Laboratory (LANL) Parallel Ocean Program (POP) model. It is a primitive equation z-level model with a free-surface boundary condition. Approximations to the governing fluid dynamics equations permit a decoupling of the model solution into barotropic (vertically averaged) and baroclinic (deviations from vertically averaged) components; these are solved using an implicit elliptic scheme and an explicit parabolic equation system, respectively. It is written in Fortran90 and was designed to run on multi-processor machines using domain decomposition in latitude and longitude. MPI is used for inter-processor communications on distributed memory machines and SHMEM on shared memory machines. Further technical details and references regarding the code and its adaptation for massively parallel computers can be obtained from http://climate.acl.lanl.gov. The sea ice model in PIPS3.0 is configured on the same grid as the ocean model. A system of ice model equations is solved as a set of coupled, initial/boundary value problems using a staggered Arakawa-B gird. Details of numerical approach in this model can be found in Bitz (2000).

Benchmarking shows POP to be highly scalable onto a large number of processors provided the processor sub-grid is large enough (Figure 1). Timings of a flat-bottom test case where the number of grid points correspond to the sizes of global grids with horizontal resolutions of 0.1°, 0.2°, and 0.4°, and 40 vertical levels were made on an IBM SP3 (Navy Oceanographic Office) using 160, 320, 500, and 600 processors. The barotropic mode is less scalable at lower resolutions and higher number of processors. In the 0.4° case the barotropic mode dominates the total run-time for all but the smallest number of processors. This is caused by too much time being spend in communication among nodes relative to that in calculation on each node. To improve performance of the barotropic model and to increase on-node performance by a factor of two numerical improvements are underway using an OpenMP/MPI hybrid scheme.

Any ocean model to be used in these coupled prediction systems must be capable of producing spatial scales between 10 and 1000 km and temporal scales up to several months. One of the challenges facing us therefore, has been the trade-off between adequate model resolution and the availability of computing resources. Through a Department of Defense (DOD) High Performance Computing Modernization Office (HPCMO) Grand Challenge Grant, we have been able to perform these very realistic simulations that hitherto have not been possible. Results presented here reflect these

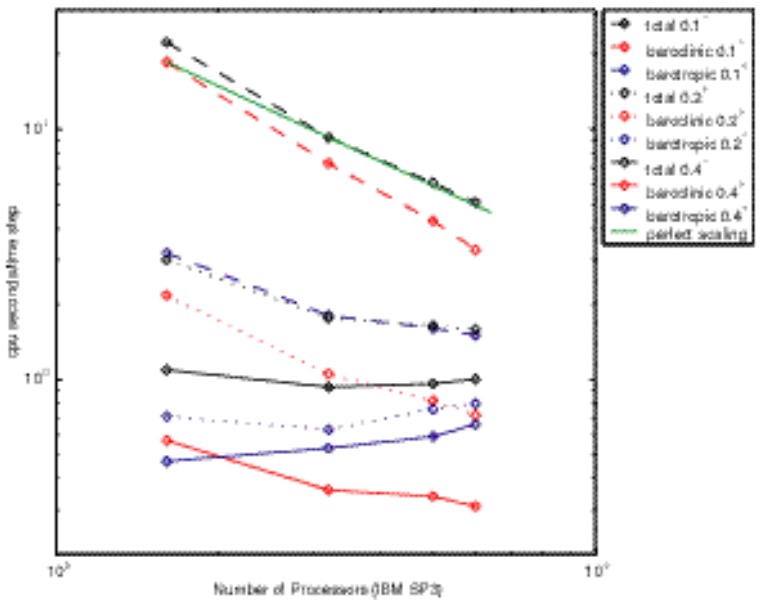


Fig. 1. Total (black), baroclinic(red), and barotropic (blue) timings (wall clock time per time step) on a log-log scale of global POP with horizontal resolutions of 0.1° , 0.2°, and 0.4°, and 40 vertical levels on an IBM SP3.

challenges and concentrate on improvements from high resolution and added physics.

2 A High-Resolution Global Ocean POP simulation

Ocean models used for Navy prediction purposes must be able to realistically reproduce the statistical nature of the surface circulation. Prior to committing to a chosen resolution for the global simulation, quantitative measures of the realism of this flow were calculated from two Mercator configurations of POP: a recent 0.1°, 40-level North Atlantic only simulation and a 0.28°, 20-level near-global case run several years ago at LANL (Maltrud et al., 1998). If the statistics from the higher resolution POP were only slightly more realistic than those from the coarser run, then large computational savings can be made using lower resolution. Both models were forced with daily winds and monthly climatological heat fluxes. An explicit mixed-layer formulation, K-profile Parameterization (KPP), was active in the North Atlantic POP.

The evaluations of the ocean models were performed using the North Atlantic

surface drifter data set for the years 1993-1997; the spatial and temporal coverage of these drifters is extensive, providing an excellent database from which to calculate statistics of the surface circulation. The drifter tracks and the numerical trajectories from the 0.28° and 0.1° POP runs from 1995 and 1996 are seen in Figures 2a, b, and c, respectively; two years only are plotted so that details can be seen clearly. Specifics of the realism of these surface trajectories are discussed further in McClean et al. (2001). Here, it is sufficient to state that the coverage of the domain by the 0.28° POP trajectories displays many gaps and is not uniform, unlike the observations. Also many of the trajectories preferentially follow coherent flows. The coverage by the 0.1° trajectories is much more extensive arising from the increased mesoscale eddy activity in the higher resolution model. Additionally, the pathways and structure of the currents are more faithfully represented in the higher resolution case. Quantitative calculations supported these qualitative findings. Eulerian results showed that flow features in the coarser run were unrealistic or misplaced and the variability was underrepresented relative to the observations; in the 0.1° POP the variability and current structures were much more realistically simulated. The intrinsic Lagrangian (trajectory-based) scales from the 0.1° POP were not statistically different from the observed quantities, while those from the 0.28° model did differ.

Based on these and other results (Bryan et al., 1998), it was decided that a global simulation to be used for synoptic forecasting would require horizontal and vertical resolutions of at least 0.1° and 40 vertical levels, respectively. Such a simulation is underway using 500 processors on the IBM SP3 at the Navy Oceanographic Office. The model uses a displaced pole grid whereby the North Pole is rotated into Hudson Bay avoiding the issue of the polar singularity. The grid consists of 3600x2400x40 grid points with 0.1° at the equator. A blended bathymetry was created from Sandwell and Smith (1997, http://topex.ucsd.edu/marine_topo), International Bathymetric Chart of the Arctic Ocean (IBCAO, Jakobsson et al., 2000), and British Antarctic Survey (BEDMAP, http://www.antarctica.ac.uk/bedmap) products. The model was initialized using the Navy's MODAS 1/8° January climatology outside of the Arctic and the University of Washington's Polar Hydrography winter climatology in the Arctic (http://psc.apl.washington.edu/Climatology.html). Surface momentum, heat, and salinity fluxes were calculated using bulk formulae based on the model surface temperature and an atmospheric state comprised of daily and monthly data from a variety of sources (as in Large et al., 1997). These fluxes are used to force the model during the current twenty-year spin-up. The KPP mixed layer formulation is active.

Figure 3 shows sea surface temperature and sea surface height fields from the Atlantic and Indian Oceans, respectively, at the end of the first year of the spin-up. Gulf Stream and equatorial fronts are clearly seen in the Atlantic and mesoscale activity

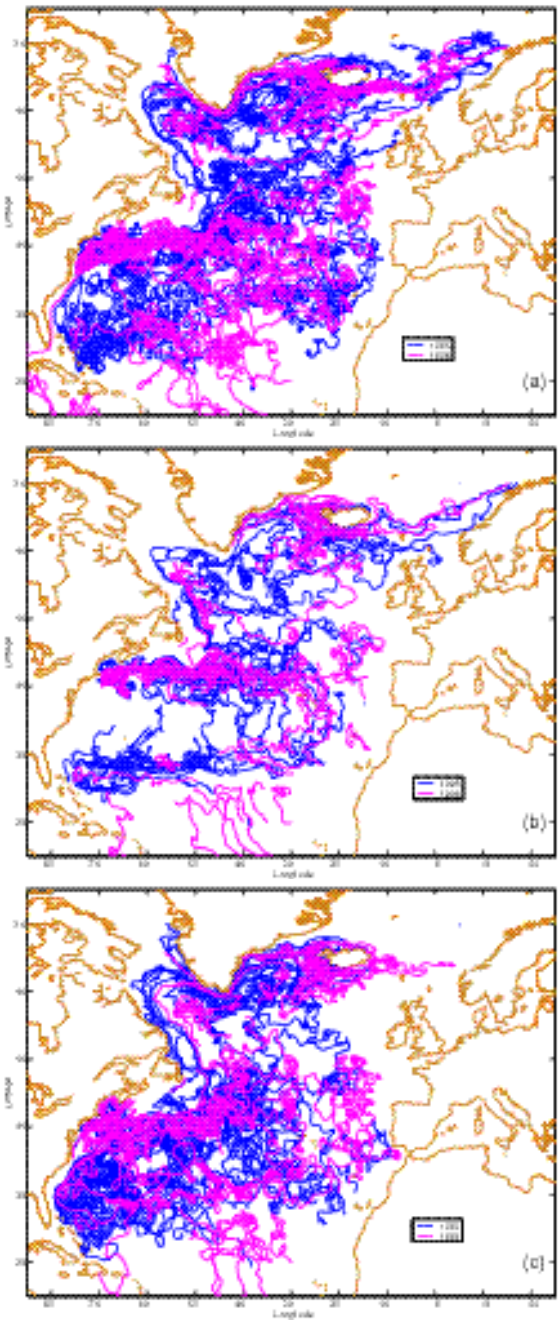


Fig. 2. North Atlantic (a) surface drifter tracks, (b) 0.28°, and (c) 0.1° POP numerical trajectories for 1995-1996

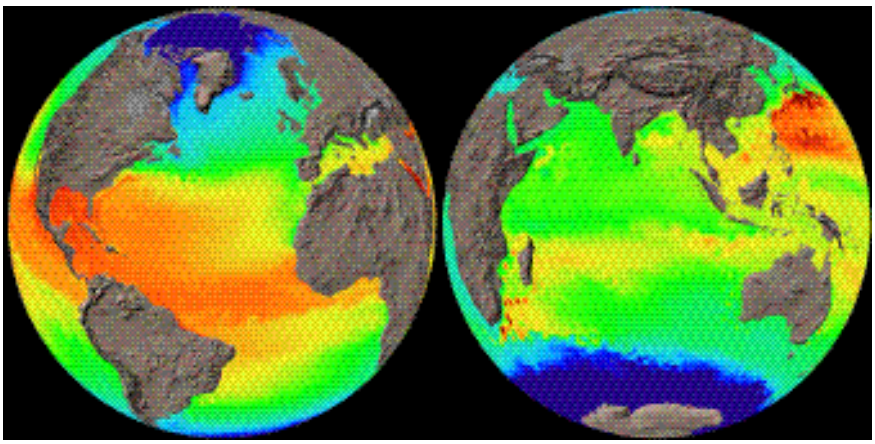


Fig. 3. Snapshots of sea surface temperature and sea surface height in the Atlantic and Indian Oceans, respectively, from the 0.1° global POP integration.

associated with the Agulhas Current offshore of east Africa is apparent. The separation of the Kuroshio off Japan is also observed. Output from the spin-up is being monitored to watch the set-up of the major currents, the impact of outflows on the thermohaline circulation, and the development of water masses. Following the spin-up, the model will be forced with realistic Navy surface forcing for the better part of a decade for the purposes of understanding features and processes important to the Navy in many different parts of the globe.

3 PIPS 3.0 Model Description and Results

The PIPS 3.0 model is configured on a 1/12° (~9 km) rotated spherical coordinate grid. The model domain (Figure 4) extends from the North Pacific at ~30°N, through the Arctic Ocean into the North Atlantic to ~40°N. The model bathymetry incorporates the 2.5-km resolution IBCAO digital bathymetry data set. It is represented by 45 z-coordinate levels. The model is considered to be eddy-permitting as features down to 40-50 km can be resolved. With the radius of deformation in the Arctic Ocean approaching 10 km, many of the smaller features are still not properly accounted for. The high resolution combined with the pan-Arctic domain allows the representation of most of the important processes in the Arctic Ocean and realistic exchanges between the North Pacific, the Arctic Ocean, and the North Atlantic.

The sea ice model at present uses viscous-plastic rheology and the zero-layer approximation of heat conduction through ice. The ongoing upgrade of this model inclu-

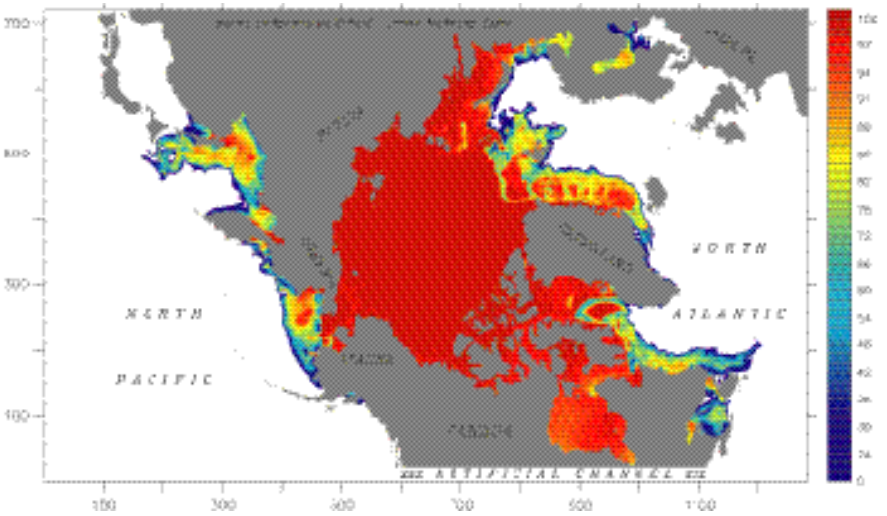


Fig. 4. A snapshot of sea ice concentration (%) from the he 9-km PIPS 3.0 model from March of year 22 of the model spin-up. The full model domain is shown.

des a Lagrangian formulation for calculating multi-category ice thickness distribution, a snow layer, a non-linear profile of temperature and salinity (Bitz 2000) and a Coulombic yield curve for the viscous-plastic rheology (Hibler and Schulson, 2000).

Animations of ice concentration (Figure 4) and thickness fields (www.oc.nps.navy.mil/~pips3) show realistic details of the annual ice structure, including oriented leads in the Western Arctic, polynyas in the Bering and Chukchi seas, and seasonal ice-edge advancement/retreat in the marginal seas of the North Pacific and the North Atlantic. The position and structure of the ice edge position in those regions appears to be significantly influenced by the ocean dynamics and water mass properties (Zhang et., 1999).

In an effort to balance the net flow of water from the Pacific Ocean into the Arctic Ocean, a 500-m deep, 162-km wide channel was created through North America connecting the Atlantic Ocean to the Pacific Ocean (Figure 4). Along the channel, westward wind forcing is prescribed at the ocean surface but otherwise the flow through the channel and through Bering Strait is not prescribed. This approach results in a net mean transport of 0.65 Sv during the model spin-up (Figure 5) which is reasonably close to the observed mean flow through the Bering Strait of 0.83 Sv.

Preliminary regional comparisons of eddy kinetic energy with our earlier 18-km version of the coupled ice-ocean model (Maslowski et al., 2000) reveal on average a tenfold increase in eddy kinetic energy in the 9-km model (Figure 6). Most importantly the large scale ocean circulation, which strongly influences the sea ice thickness

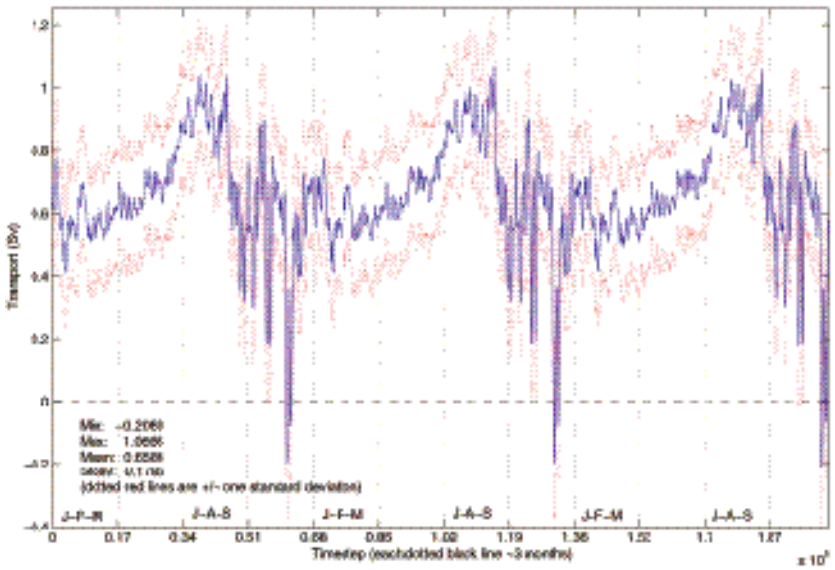


Fig. 5. The net volume transport (1 Sv = 10^6 m^3s^{-1}) through the Bering Strait from the years 24-26 of the model spin-up.

and concentration especially in marginal ice zones, is properly represented in this model. The narrow boundary currents associated with the continental margins of the deep central Arctic Ocean are only 100-150 km wide but they are believed to be the main sources of heat and salt advected northward from the North Atlantic. These predominantly barotropic flows are by definition strongly dependent on bathymetry (e.g. shelf slopes and submarine ridges), which provides another argument for using high resolution to resolve details of the bottom topography and boundary current flows.

One of the pathways of Atlantic Water transport into the Arctic Ocean includes the flow through the Barents Sea. We have analyzed monthly, seasonal and annual volume and property transports through the Barents Sea in order to evaluate model results in comparison with observations from this region. The calculated fluxes depend crucially on the inflow of heat and salt from the Norwegian Sea via the North Cape Current and on seasonal ice melt and growth in the Barents Sea. The modified Atlantic Water leaves the region primarily through the St. Anna Trough and it significantly affects Arctic Ocean water mass structure. The model realistically simulates known circulation and water mass characteristics as well as the seasonally dependent ice edge position in the Barents Sea. Results indicate an annual volume transport of 3.9 Sv and heat transport of 95TW into the Barents Sea, between Svalbard and Norway. Annual average volume and heat transport into the Arctic Ocean, between Franz Jo-

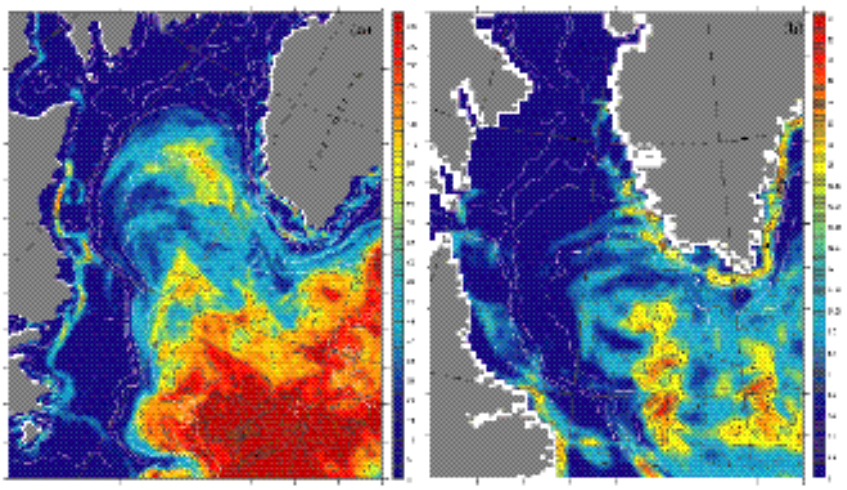


Fig. 6. The surface eddy kinetic energy (cm^2s^{-2}) in the Labrador Sea calculated from (a) the 9-km model at depths 0-5 m for the year 13 of spin-up and (b) the 18-km model at depths 0-20 m for the year 1997.

seph Land and Novaya Zemlya are 3.2 Sv and 16.6 TW, respectively. The magnitudes of the model transports agree well with observations. Continued integration using 1979-1999 daily varying interannual forcing will allow model-data comparison of interannual variability, including possible trends or regime shifts in response to large scale changes in the atmospheric weather patterns.

Conclusions

Results from two high resolution modeling efforts are presented to demonstrate the importance of high resolution in simulating the ocean and sea-ice circulation. Details of the 0.1°, 40-level global POP spin-up were provided along with initial results that showed the model to be realistically simulating surface frontal structures and mesoscale activity, both of which are very important to Navy prediction needs. In PIPS 3.0, mean volume and heat transports associated with the main pathways into the Arctic were found to be realistic compared with observations. Improvements to the sea ice model are likely to produce both more realistic ice configurations and seasonal-to-interannual variability in the multi-year ice cover.

Acknowledgements

Funding was provided by the Office of Naval Research, the National Science Foundation, and the Department of Energy (CCPP program). The simulations were performed at the Army Research Laboratory, the Navy Oceanographic Office, the Arctic Region Supercomputing Center, and the Advanced Computer Laboratory at LANL. The drifter data was provided by the Atlantic Oceanographic and Meteorological Laboratory Drifting Buoy Data Assembly Center. Pam Posey and Steve Piacsek (both NRL) supplied the Navy winds and the initial condition, respectively. Collaborative work by Pierre Poulain and Jimmy Pelton on the drifter/model studies, and Doug Marble (all NPS) on the 9-km model are acknowledged.

References

1. Bitz, C.M., 2000: Documentation of a Lagrangian sea ice thickness distribution model with energy-conserving thermodynamics, APL-UW TM 8-00, 49 pp. University of Washington, Seattle, WA.
2. Bryan, F. O., R. D. Smith, M. E. Maltrud, and M. W. Hecht, 1998: Modeling the North Atlantic Circulation: From eddy permitting to eddy resolving. WOCE International Conference, Halifax, Nova Scotia.
3. Hibler, III, W. D., and E. M. Schulson, 2000: On modeling the anisotropic failure and flow of flawed sea ice, J. Geophys. Res., 105 (C7), 17,105-17,120.
4. Jakobsson, M., N. Z. Cherkis, J. Woodward, R. Macnab, and B. Coakley, 2000: New grid of Arctic bathymetry aids scientists and mapmakers, EOS Trans., Am. Geophys. Union, 81 (9).
5. Large, W. G., G. Danabasoglu, S. C. Doney, and J. C. Williams, 1997: Sensitivity to surface forcing and boundary layer parameterization. Rev. Geophys, 32, 363-404.
6. Maltrud, M. E., and R. D. Smith, A. J. Semtner, and R. C. Malone, 1998: Global eddy-resolving ocean simulations driven by 1985-1995 atmospheric winds. J. Geophys. Res., 103, 30825-30853.
7. Maslowski, W., B. Newton, P. Schlosser, A. Semtner, and D. Martinson, 2000: Modeling recent climate variability in the Arctic Ocean, Geophys. Res. Lett., 27(22), 3743-3746.
8. McClean, J. L., P.-M. Poulain, J.W. Pelton, and M. E. Maltrud, 2001: Eulerian and Lagrangian statistics from surface drifters and two POP models in the North Atlantic, J. Phys. Oceanogr., submitted.
9. Smith, W.H.F., and D.T. Sandwell, 1997: Global sea floor topography from satellite altimetry and ship-depth soundings, Science, 277, 1957-1962.
10. Zhang, Y., W. Maslowski, and A. J. Semtner, 1999: Impact of mesoscale ocean currents on sea ice in high-resolution Arctic ice and ocean simulations, J. Geophys. Res., 104 (C8), 18,409-18429.

New Materials Design

Jerry Boatz[1], Mark S. Gordon[2], Gregory Voth[3], Sharon Hammes-Schiffer[4], and Ruth Pachter[6]

[1] Air Force Research Laboratory, AFRL/PRSP, Edwards AFB, CA 93524
[2] Department of Chemistry, Iowa State University, Ames, IA 50011
[3] Department of Chemistry and Henry Eyring Center for Theoretical Chemistry, University of Utah, Salt Lake City, UT 84112
[4] Department of Chemistry, The Pennsylvania State University, University Park, PA 16802
[5] Air Force Research Laboratory, AFRL/MLPJ, Wright-Patterson AFB, OH 45433

Abstract. In this paper we report our recent results on the design of materials with controlled properties by the application of computational chemistry methods, new algorithms and scalable software.

1 Introduction

Developments of new materials, such as new high-energy POSS (polyhedral oligomeric silsesquioxanes), that are highly resistant to extreme environments and are therefore desirable coatings for rocket engines, nonlinear optical materials, and liquid crystals, are important in many applications of interest to the Air Force. However, the design of large and complex material systems with controlled properties is challenging. Therefore, the calculations we describe in this paper would not have been possible without scalable quantum chemistry codes such as GAMESS (General Atomic and Molecular Electronic Structure System), developed under the auspices of a DoD CHSSI (Computational High Performance Scalable Software Initiative) grant, Gaussian, and molecular dynamics codes, as well as the availability of the DoD Computation Centers through our computational challenge project.

2 Results and Discussion

2.1 High-Energy Density Materials (HEDM)

2.1.1 Bond Dissociation Energies: A Comparative Study of Theoretical Methods[1,2]

The purpose of this study is to perform a systematic comparison of several theoretical methods for computing the bond dissociation energies (BDEs) of a prototypical energetic compound. The computational methods include single-reference (Moller-Plesset second order perturbation theory (MP2)[1], Gaussian-2 using MP2 (G2(MP2))[2], and coupled cluster with single and double substitutions (CCSD(T))[3]), multiconfigurational (complete active space self-consistent field

---

1 This work was done in collaboration with Donald L. Thompson and Dan C. Sorescu, Department of Chemistry, Oklahoma State University, Stillwater, OK 74078.

2 We thank Dr. Jeffrey A. Sheehy for several stimulating discussions.

V.N. Alexandrov et al. (Eds.): ICCS 2001, LNCS 2073, pp. 1108–1116, 2001.
© Springer-Verlag Berlin Heidelberg 2001

(CASSCF)[4] and multiconfiguration quasi-degenerate perturbation theory (MCQDPT)[5]), as well as density functional theory (DFT, using Becke's three parameter hybrid exchange functional with the Lee, Yang, and Parr correlation (B3LYP)[6]) methods. The energetic compounds of interest are 1,1-diamino-2,2-dinitroethylene (FOX-7) and its simpler analogues. FOX-7 was recently synthesized[7] and found to have superior impact and friction sensitivity properties relative to the well-known HMX and RDX explosives. This molecular system is an example of a "push-pull" ethylene compound, which has both electron donating (NH_2) and electron withdrawing (NO_2) groups attached to a common ethylene linkage. Such compounds are potentially stabilized through "π" resonance delocalization and hence may possess non-negligible multiconfigurational character. Therefore, this molecule should be a good testbed for comparing DFT and single-configuration methods against multiconfigurational techniques.

The geometries of the reactants and products of the C-NH_2 and C-NO_2 homolytic bond dissociation reactions of FOX-7 and its analogues have been fully optimized and verified as local minima at the B3LYP, MP2 and CASSCF levels, using the 6-311G(d,p) basis set.[8] CCSD(T)//MP2, MCQDPT//CASSCF, and G2(MP2) energies were also computed. The putative resonance delocalization of the "π" electrons in FOX-7 suggests that in addition to the (2e,2o) C-N σ,σ^* space needed to describe homolytic bond cleavage, an appropriate active space for the MCSCF calculations should include the (2e,2o) C-C$\pi+\pi^*$, (4e,3o) NO_2 π+non-bonding+π^*, and (2e,1o) NH_2 lone pair electrons and orbitals, described in further detail elsewhere.[9] The degree of multiconfigurational character is assessed by calculation of the natural orbital occupation numbers (NOONs) of the MP2, CCSD(T), and CASSCF first order density matrices. As shown in a recent work by Gordon[10] et al., the occurrence of "non-physical" MP2 and CCSD(T) NOONs is an indication that a single configuration treatment may be inadequate. All calculations were performed using GAMESS[11], GAUSSIAN98[12], and ACESII.[13]

At the MP2 level, aminoethylene is the only closed shell compound without at least one non-physical NOON. Although some calculations are still in progress, it appears that all open-shell species except the amino and nitro radicals have multiple non-physical NOONs at this level of theory. In contrast, none of the compounds have non-physical NOONs at the CCSD(T) level, although several open shell CCSD(T) calculations are still in progress. This indicates that CCSD(T) is able to capture more of the non-dynamical correlation than MP2 and hence should be more reliable. The CASSCF NOONs show that the lone pair orbital on the NH_2 group has little multiconfigurational character. However, the nominally unoccupied NO_2 π^* (0.08–0.14 electrons) orbitals have relatively large populations, suggesting that the C-NO_2 BDEs may not be as well described by single reference methods as the C-NH_2 BDEs.

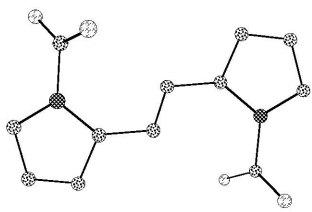
The predicted C-NH_2 BDEs for aminoethylene (which include zero-point energy corrections) are all within 9 kcal/mol of the experimental value of 102 kcal/mol[14] and range from 93 (MCQDPT) to 108 (MP2) kcal/mol. The B3P86/6-31+G(d,p)[15] and G2(MP2) predictions are both within about 1 kcal/mol of experiment. MP2 overestimates the BDE by 6 kcal/mol and MCQDPT underestimates it by about 9 kcal/mol, with CCSD(T) (97.4 kcal/mol) and B3LYP (96.7) yielding intemediate values. CCSD(T) is the single reference method in closest agreement with MCQDPT.

For 1-amino-2-nitroethylene, MP2 again gives the largest C-NH_2 BDE (126 kcal/mol) and MCQDPT the smallest (112 kcal/mol). CCSD(T) is again the single-reference method in closest agreement with MCQDPT. Although the C-NH_2 BDE

calculations for the entire set of ethylene derivatives are incomplete, it appears that (a) MP2 consistently gives the highest C-NH_2 BDEs and (b) the MCQPDT BDEs are in good agreement with CCSD(T). In contrast, the DFT BDEs show no clear patterns of behavior. For nitroethylene, the C-NO_2 BDEs range from 59.5 kcal/mol (MCQDPT) to 79.4 kcal/mol (G2(MP2)). CCSD(T) is the single-reference method, which is in closest agreement with MCQDPT, although the difference between these two values is greater than in the case of the C-NH_2 BDEs of aminoethylene. The B3LYP value of 68.8 kcal/mol is in good agreement with CCSD(T) (70.9 kcal/mol), although the B3P86 prediction of 74.5 kcal/mol[15] is closest to MP2 (77.7 kcal/mol). Similar observations apply to the predicted C-NO_2 bond energies in 1-amino-2-nitroethylene, which range from 69.3 (MCQDPT) to 88.8 kcal/mol (G2(MP2)). CCSD(T) (78.2 kcal/mol) is the single-reference method in closest agreement with the MCQDPT result and also is in good agreement with the B3LYP value (81.4 kcal/mol). MCQDPT also predicts the smallest C-NO_2 BDE for 1-amino-2-nitroethylene (69 kcal/mol). Although the C-NO_2 BDE calculations are not yet complete, it appears that (a) G2(MP2) and MP2 consistently give the highest C-NO_2 BDEs and (b) the single reference method in best agreement with MCQDPT is CCSD(T).

2.1.2 High-Energy Density Materials

A recent major focus centers around a series of proposed monopropellants.16], such

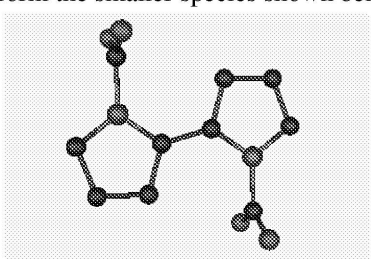


as:

I_{sp}, which is a measure of the energy released by the reaction of the fuel divided by the mass of the products, is of primary interest:

$$I_{sp} \propto [\Delta H/m]^{1/2} \quad (1)$$

Using a combination of isodesmic reactions and the G2 model, the heat of formation for this species is predicted to be 456.8 kcal/mol. This translates to an I_{sp} of 240 sec, compared with 230 sec for hydrazine, thus being a very promising fuel. However, it is important to consider the stability of such high-energy species to various reactions, before asserting their viability as fuels. One possible reaction is the loss of molecular nitrogen to form the smaller species shown below:



Using our highly scalable second order perturbation theory code with a large basis set, we have found that this decomposition process is exothermic by 60 kcal/mol. It

may be that the barrier for decomposition is large, in which case the species may still be viable. The search for the transition state for this reaction is in progress.

Recently the first new all-nitrogen compound in nearly a century N_5^+[17] was synthesized. The counterion for this species was AsF_6^-, the result being a rather unstable solid. In order to understand the potential energy surface for this species, we performed MP2 calculations on the various stationary points. The first step in this process was to separate the two ions by a large distance and optimize the geometry. The result was an ion pair $[AsF_6^-][N_5^+]$ complex that is, not surprisingly, 98 kcal/mol below the separated ions. What was somewhat surprising is that this complex is 47 kcal/mol below the neutral species that is formed when one F^- ion is transferred to N_5^+ to form neutral $AsF_5 + FN_5$. The former is a well-known species, but the latter has not previously been reported. FN_5 has several isomers, whose relative energies and the barriers separating them were determined using large basis set CCSD(T) calculations. The lowest lying isomers were found to have small energy barriers (< 10 kcal/mol) separating them from the much more stable decomposition products $FN_3 + N_2$. The exothermicity of this decomposition is more than 50 kcal/mol, so the products $AsF_5 + FN_3 + N_2$ are slightly lower in energy than the original ion pair $[AsF_6^-][N_5^+]$. Since FN_3 is itself an energetic species with a relatively small barrier to further decomposition, it is therefore not surprising that the original species made by Christe et al. is unstable.

2.1.3 Quantum Molecular Dynamics (MD) Simulations of Low Temperature HEDM

Traditional MD algorithms can be extended to incorporate equilibrium quantum mechanical effects through the use of discretized Feynman path integrals[18]. The resulting algorithm, path integral MD (PIMD), is highly amenable to parallelization because each physical particle becomes rigorously mapped onto a collection of classical-like quasiparticles at different values of imaginary time, each connected to its neighbor by a harmonic-like force. The discretized PIMD algorithm can be load-balanced across computational nodes because the quasiparticles with the same imaginary time index experience identical forces. The inter-node communications are minimal since there are only nearest-neighbor harmonic forces to compute. The PIMD algorithm, no matter how complex the physical system, can always be made to achieve excellent scalability and thus high performance. Moreover, classical MD techniques such as constant temperature and pressure algorithms can be readily incorporated into PIMD. The algorithm is even amenable to a two-tier level of parallelism, the first being over the imaginary time slices (quasiparticles), the second over the force loop for the interparticle interactions in large (thousand or even million) particle simulations.

In an important theoretical advance[19,20,21,22], PIMD methods have been extended to include quantum dynamical effects (i.e., not just equilibrium properties). This computational approach is called "Centroid Molecular Dynamics" (CMD). It incorporates the dominant quantum dynamical effects of a many-body system into a classical-like MD framework, thus significantly extending the range and applicability of MD methods. The basic result of CMD[19-22] is that the dynamical correlations of quantum particles in a general many-body system can be accurately computed by running classical-like trajectories on an effective potential which includes the effects of quantum zero point energy and tunneling. The task of integrating the CMD equations is not trivial since the effective potential is a quantum potential of mean force, requiring an "on the fly" dynamical averaging at each timestep. A number of powerful algorithms for solving the CMD equation have been developed[22]. The best CMD algorithm so far is the “hyperparallel” CMD approach[23] using an

adiabatic algorithm[22] in the first tier of parallelism to determine the effective potential; a second tier of parallelism can be introduced in which, for each of the averaging slaves, the MD steps are sent to additional nodes for each discretized path integral imaginary time slice[23]. A third tier of parallelism for very large systems for each imaginary time slice, the classical force loop is split over several nodes. The scaling of the hyperparallel CMD code on the IBM SP and SGI O2000, as well as a Pentium Linux cluster, is excellent (Fig. 1). The parallelization of the long rage forces (Ewald summation) was also achieved.

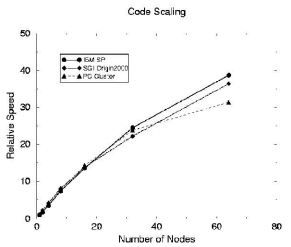


Fig. 1. Scaling of CMD

The PIMD and CMD algorithms are of general interest in all areas of condensed matter computer simulation. However, they are particularly important for the HEDM program in the computational study of, e.g., solid hydrogen, helium, or nitrogen matrices which contain isolated reactive species such as lithium, boron, aluminum, hydrogen atoms, organic radicals, etc. Such matrices are high priority targets as possible HEDMs for Air Force space propulsion purposes. At the temperatures appropriate to the condensed phases of hydrogen or helium, the host matrix molecules will exhibit large quantum mechanical effects, but the treatment of these effects by a direct attack on the time-dependent Schrödinger equation is impossible. The key issue is the stability of the HEDM (i.e., the ability to trap the energetic impurities for some period of time and thus impede their diffusion and eventual recombination). Both the structural and dynamical aspects of the composite condensed phase HEDM systems are of interest. For the equilibrium structural studies, PIMD methods are adequate. For the dynamical studies, the CMD approach must be employed to directly simulate the diffusive and recombination steps of the guest atoms in the solid host. The long-term goal of this research is to develop general computational methods to rapidly and efficiently characterize proposed high energy density materials.

The structure and stability of atomic impurities trapped in solid *para*-hydrogen have been studied by employing large scale PIMD and CMD simulations at 4 K and zero external pressure. Starting from pure solid hydrogen, doped systems were prepared by substituting impurity atoms for hydrogen molecules at substitutional defect sites. The structural and thermodynamic properties of the resulting HEDM were then calculated. In the case of atomic boron impurities, the impurity atom interacts anisotropically with the matrix *para*-hydrogen molecules because of its singly filled 2p orbital. To assess the effect of the electronic anisotropy in the potential, a comparison has been made between simulations in which the orientation-dependent (anisotropic) and orientation-averaged (spherical) B–H_2 potentials were used. In collaboration with the Alexander group[24], three matrices were investigated: (a) one with a single B atom site-substituted for a *p*-H_2 molecule, (b) one with a similar site-substituted matrix having a nearest-neighbor vacancy, (c) and one with a periodic "super cell" simulation box having two hydrogen bulk slab regions in order to study a B impurity near the *p*-H_2 surface.

It was found that significant "second-order" distortions of the lattice occur to permit an energetically favorable orientation of the 2p orbital, even in the absence of a neighboring vacancy. Especially when the B impurity is located near the surface, the spherically averaged potential provides a noticeably different description from the case of the anisotropic potential. These quantum MD simulations reveal the large structural differences that can be found in the lattice resulting from the anisotropic boron-hydrogen interaction potential. The implications of this behavior will be explored in future research.

2.2 POSS Compounds

There is great interest in POSS compounds because of their resistance to extreme environments and the ease with which they are synthesized. Since almost nothing is known experimentally about the mechanism by which they are formed, we have embarked on a long term project to determine the possible competing mechanisms as a function of catalyst, solvent, and substituents (e.g., in[25]). The synthesis begins with hydrolysis of $RSiX_3$ to $RSi(OH)_3$, followed by condensation to the siloxane $RSi(OH)_2$-O-$Si(OH)_2R$. Subsequent condensations lead to the 3D cage compounds. We have shown that the initial hydrolsysis and condensation steps all have very high-energy barriers in the gas phase, but are being reduced to nearly zero by the presence of one water molecule added to represent the solvent. Thus, all steps leading to the initial disiloxanes and to the ring compounds D_3 and D_4 occur with net energy requirements of less than 10 kcal/mol. In this paper we describe current efforts to (a) determine substituent effects on these barriers, (b) determine the effects of adding additional waters, and (c) to compare the properties of the 3D POSS compounds with their Ti analogs. The substituents studied include R=H, methyl, t-butyl and phenyl, and X=Cl, OCH_3. There is experimental interest in synthesizing incompletely condensed POSS. Because the reaction is very fast, it always goes to completion. The interest in substituent effects lies in attempting to slow down the reaction, possibly using bulky R groups. There is industrial interest in the possibility that Ti POSS may be another class of materials with desirable properties. Hammes-Schiffer et al.[26] have been developing methodologies for the simulation of hydrogen transfer reactions, and most recently a new molecular orbital method for the simultaneous calculation of electronic and hydrogen vibrational wavefunctions, now incorporated into GAMESS. This method may be used to obtain minimum energy reaction paths and direct dynamics trajectories of hydrogen transfer reactions, with the advantage that nuclear quantum effects such as zero point energy and hydrogen tunneling are incorporated during the generation of the reaction paths and trajectories, rather than subsequently included as corrections. This approach will be combined with a mixed quantum/classical surface hopping method to study the quantum dynamics of hydrogen transfer reactions with an *ab initio* potential energy surface obtained "on the fly," and used to investigate hydrogen transfer reactions in the hydrolysis and condensation steps required for the synthesis of POSS, in order to determine the influence of different trihalosilane reactants on the rates and yields of these steps and aid in the efficient synthesis of POSS.

2.3 Nonlinear Absorbing Materials

We have previously shown that density functional theory and the time-dependent density functional response theory (TDDFT) (B3LYP) accurately predict the structures and spectra of porphin and the larger tetraphenylporphyrin and its β-

octahalogenated derivatives,[27],28] thus enabling their assessment as reverse saturable absorbing materials For PH_2, the photoinduced tautomerization process has also been investigated[29] in order to discern between previously proposed discordant mechanisms. Recently we carried out calculations[30] for tetrabenzoporphyrin (ZnTBP) and phthalocyanine (ZnPc) (Fig. 2).

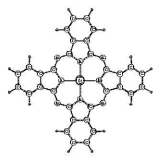


Fig. 2. Zinc phthalocyanine

Our calculations support the multiple bands in the B and higher energy regions of the ZnPc and ZnTBP spectra that have been determined from experiment. Despite considerable experimental and theoretical work on ZnPc and ZnTBP, their ground state electronic structures and UV spectra are not well understood. Previous theoretical studies provided only a qualitative understanding in the low-energy region, and none for the higher energy bands. We have elucidated significant shifts in Gouterman's four orbitals of ZnP upon tetraazasubstitution and tetrabenzoannulation (detailed in[30]). The benzo groups destabilize the $1a_{1u}$ orbital in ZnP, but they do not show a significant deviation from the four-orbital model for the interpretation of the Q and B bands of ZnTBP. However, tetraazasubstitions lower considerably the $4a_{2u}$ orbital (HOMO-1) in ZnTBP. Also, it is shown that the near degeneracy of the HOMO and HOMO-1 in ZnP that provides the basis for Gouterman's four orbital model does not hold for ZnPc. In ZnPc, the near degeneracy of the HOMO-1 ($4a_{2u}$) with other occupied orbitals gives rise to a complex structure of the higher energy regions of the spectrum that were not reproduced by simple MO methods. The $4a_{2u}$ MO has significant π contributions from the benzo rings in ZnPc but not in ZnTBP. The Q and B bands are predicted to appear at 2.18 and 3.28 eV for ZnTBP, respectively, in excellent agreement with the maxima located at 2.02 and 3.18 eV as established from the absorption spectrum in a supersonic jet expansion, with similar values (2.02 & 3.10 eV) obtained from the absorption spectrum in an Ar matrix. Tetraazasubstitutions are predicted to redshift the Q band of ZnTBP by about 0.1 eV. The computed excitation energies in near quantitative agreement with experiment clearly provide a strong basis for the interpretation of the electronic spectra of such large molecular systems. TDDFT calculations are also being carried out for candidate two-photon absorption (TPA) materials,[31] to be compared to those of Albota et al.[32]

2.4 Liquid Crystals

Recent work is in progress for studying a liquid crystalline droplet to model the behavior of a liquid crystal in the bulk, with the details reported elsewhere.[33] Indeed, atomic level simulations are proven important in understanding the structure-property relations of materials. Due to the size of the LC droplet very large-scale classical MD simulations of such systems are required, that may, become computationally prohibitive. The fast multipole method (FMM)[34], which uses a multiscale hierarchy of partitions of the volume and a divide-and-conquer strategy to compute the power series, allows all the forces to be computed to any specified

accuracy in O(N) operations. FMM3D (an implementation and improvement to FMM in 3-D), which contains a variety of schemes for computing multipole translations, has been implemented into the MD program that is being developed in our group.[35]

3 References

1. Pople, J.A., Binkley, J.S., Seeger: R., Int. J. Quantum Chem., 10 (1976) 1; Bartlett, R.J., Silver, D.M.: Int. J. Quantum Chem., 9 (1975) 183; Dupuis, M., Chin, S., Marquez, A.: In: Relativistic and Electron Correlation Effects in Molecules, Malli, G.: Ed. Plenum Press, NY 1994; Frisch, M.J., Head-Gordon, M., Pople, J.A.: Chem. Phys. Lett. 1660 (1990) 275-280
2. Curtiss, L.A., Raghavachari, K., Pople, J.A.: J. Chem. Phys. 98 (1993) 1293
3. See, for example, Bartlett, R.J., Stanton, J.F.: In: Applications of Post-Hartree-Fock Methods: A Tutorial, Reviews of Computational Chemistry, Vol. V, Lipkowitz, K.B. and Boyd, D.B., eds., VCH Publishers, Inc. New York, 1994
4. Siegbahn, P.E., Heiberg, A., Roos, B.O., Levy, B.: Phys. Scr. 21 (1980) 232. b) Roos, B.O., Taylor, P.R., Siegbahn, P.E.: Chem. Phys. 48 (1980) 157. c) Roos, B.O.: Int. J. Quantum Chem. S14 (1980) 175
5. Nakano, H.: J. Chem. Phys. 99 (1993) 7983-7992
6. Becke, A.D.: J. Chem. Phys. 98 (1993) 5648
7. Bemm, U., Östmark, H.: Acta Cryst. C54 (1998) 1997; Östmark, H., Langlet, A., Bergman, H., Wellmar, U., Bemm, U.: Eleventh Int. Detonation Symp., 30 Aug – 4 Sep 1998, Snowmass Village, Colorado, USA
8. Krishnan, R., Binkley, J.S., Seeger, R., Pople, J.A.: J. Chem. Phys. 72 (1980) 650. (Six-component Cartesian d functions used.)
9. Boatz, J.A., Sorescu, D.C., Thompson, D.L.: manuscript in preparation
10. Gordon, M.S., Schmidt, M.W., Chaban, G.M., Glaesemann, K.R., Stephens, W.J., Gonzales, C.: J. Chem. Phys. 110 (1999) 4199
11. Schmidt, M.W., Baldridge, K.K., Boatz, J.A., Elbert, S.T., Gordon, M.S., Jensen, J.J., Koseki, S., Matsunaga, N., Nguyen, K.A., Su, S., Windus, T.L., Dupuis, M., Montgomery, J.A.: J. Comput. Chem. 14 (1993) 1347.
12. Gaussian 98, Revision A.7, Frisch, M. J., Trucks, G.W., Schlegel, H.B., Scuseria, Robb, M.A., Cheeseman, J.R., Zakrzewski, V.G., Montgomery, J.A., Jr., Stratmann, R.E., Burant, J.C., Dapprich, S., Millam, J.M., Daniels, A.D., Kudin, K.N., Strain, M.C., Farkas, O., Tomasi, J., Barone, V., Cossi, M., Cammi, R., Mennucci, B., Pomelli, C., Adamo, C., Clifford, S., Ochterski, J., Petersson, G.A., Ayala, P.Y., Cui, Q., Morokuma, K., Malick, D.K., Rabuck, A.D., Raghavachari, K., Foresman, J.B., Cioslowski, J., Ortiz, J.V., Baboul, A.G., Stefanov, B.B., Liu, G., Liashenko, A., Piskorz, P., Komaromi, I., Gomperts, R., Martin, R.L., Fox, D.J., Keith, T., Al-Laham, M.A., Peng, C.Y., Nanayakkara, A., Gonzalez, C., Challacombe, M., Gill, P.M.W., Johnson, B., Chen, W., Wong, M.W., Andres, J.L., Gonzalez, C., Head-Gordon, M., Replogle, E.S., and Pople, J.A.: Gaussian, Inc., Pittsburgh PA, 1998
13. Stanton, J.F., Gauss, J., Watts, J.D., Lauderdale, W.J., and Bartlett, R.D.: Int. J. Quantum Chem., S26 (1992) 879
14. Lias, S.G., Bartmess, J.E., Liebman, J.F., Holmes, J.L., Levin, R.D., and Mallard, W.G: J. Phys. Chem. Ref. Data 17 (Suppl. 1) (1988) 1-872
15. Politzer, P., Concha, M.C., Grice, M.E., Murray, J.S., Lane, P., Habibollazadeh, D.: Theochem, 452 (1998) 75
16. Work of the late Rob Schmitt & Jeff Botaro, SRI
17. Christe and co-workers, AFRL/PR
18. See, e.g., Berne, B.J., and Thirumalai, D.: Annu. Rev. Phys. Chem. 37 (1987) 401
19. Cao, J. and Voth, G.A.: J. Chem. Phys. 99 (1993) 10070
20. Cao, J. and Voth, G.A.: J. Chem. Phys. 100 (1994) 5106
21. Cao, J. and Voth, G.A.: J. Chem. Phys. 101 (1994) 6157

22. Cao, J. and Voth, G.A.: J. Chem. Phys. 101 (1994) 6168
23. Calhoun, A., Pavese, M., and Voth, G.A.: Chem. Phys. Lett. 262 (1996) 415
24. Krumrine, J.R., Jang, S., Alexander, M.H., and Voth, G.A.: J. Chem. Phys. 113 (2000) 9079
25. Kudo, T., Takako, and Gordon, M.S.: J. Phys. Chem. A, 104 (2000) 4058
26. S. Hammes-Schiffer: J. Phys. Chem. A 102 (1998) 10443; Webb, S.P., Agarwal, P.K., and Hammes-Schiffer, S.: J. Phys. Chem. B 104 (2000) 888; WebbS.P., and Hammes-Schiffer, S.: J. Chem. Phys. 113 (2000) 5214
27. Nguyen, K. A., Day, P. N., and Pachter, R.: J. Chem. Phys,. 110 (1999) 9135; Nguyen, K. A., Day, P. N., and Pachter, R.: J. Phys. Chem. A, 103 (1999) 7378
28. Nguyen, K. A., Day, P. N., and Pachter, R.: J. Phys. Chem., 104 (2000) 4755
29. Nguyen, K. A. and Pachter, R., J. Phys. Chem., 104 (2000) 4549
30. Nguyen, K.A., and R. Pachter, J. Chem. Phys., manuscript in preparation
31. Day, P.N., and Pachter, R., work in progress
32. Albota, M., Beljonne, D., Bredas, J.L., Ehrlich, J.E., Fu, J.Y., Heikal, A.A., Hess, S.E., Kogej, T., Levin, M.D., Marder, S.R., McCord-Maughon, D., Perry, J.W., Rockel, H., Rumi, M., Subramaniam, G., Webb, W.W., Wu, X.L., and Xu, C.: Science 281 (1998) 1653
33. Wang, Z., Patnaik, S., and Pachter, R.: Computational and Theoretical Polymer Science, to appear, 2001
34. Greengard, L., and Rokhlin, V.: J. Comp. Phys., 73 (1987) 325
35. Wang, Z., Lupo, J.A., McKenney, A., and Pachter, R.: Proceedings of SC99 (1999)

Parallelization of an Adaptive Mesh Refinement Method for Low Mach Number Combustion*

Charles A. Rendleman, Vincent E. Beckner, and Mike Lijewski

Center for Computational Sciences and Engineering
Ernest Orlando Lawrence Berkeley Laboratory
Berkeley, CA, 94720
CARendleman@lbl.gov

Abstract. We describe the parallelization of a computer program for the adaptive mesh refinement simulation of variable density, viscous, incompressible fluid flows for low Mach number combustion. The adaptive methodology is based on the use of local grids superimposed on a coarse grid to achieve sufficient resolution in the solution. The key elements of the approach to parallelization are a dynamic load-balancing technique to distribute work to processors and a software methodology for managing data distribution and communications. The methodology is based on a message-passing model that exploits the coarse-grained parallelism inherent in the algorithms. A method is presented for parallelizing weakly sequential loops—loops with sparse dependencies among iterations.

1 Introduction

Advanced, higher-order finite difference methods and local adaptive mesh refinement have proven to be an effective combination of tools for modeling problems in fluid dynamics. However, the dynamic nature of the adaptivity in time dependent simulations makes it considerably more difficult to implement this type of methodology on modern parallel computers, particularly, distributed memory architectures. In this paper we present the parallelization of a computer program using a software framework that facilitates the development of adaptive algorithms for multiple-instruction, multiple-data (MIMD) architectures. The particular form of adaptivity we consider is a block-structured style of refinement, referred to as AMR, that was originally developed by Berger and Oliger [5]. The methodology uses the approach developed by Berger and Colella [4] for general systems of conservation laws, its extension to three dimensions by Bell et al. [2], and incompressible flows by Almgren et al. [1]. Subsequently, this work was extended to the simulation of low Mach number combustions by Pember

* This work was carried out at the Lawrence Berkeley National Laboratory under the auspices of the US Department of Energy Contract No. DE-AC03-76SF00098. Support was provided by the Defense Threat Reduction Agency under subcontract to Lawrence Livermore National Laboratory and by the Applied Mathematics Program in the DOE Office of Science.

V.N. Alexandrov et al. (Eds.): ICCS 2001, LNCS 2073, pp. 1117–1126, 2001.
© Springer-Verlag Berlin Heidelberg 2001

et al. [13] and by Day and Bell [10]. This paper discusses the AMR parallel implementation of the algorithm described by Day and Bell [10]

AMR is based on a sequence of nested grids with finer and finer mesh spacing in space, each level being advanced in time with time step intervals determined by the Courant-Friedrich-Level (CFL) condition. The fine grids are recursively embedded in coarser grids until the solution is sufficiently resolved. An error estimation procedure automatically determines the accuracy of the solution and grid management procedures dynamically create rectangular fine grids where required to maintain accuracy or remove rectangular fine grids that are no longer required for accuracy. Special difference equations are used at the interface between coarse and fine grids to insure conservation.

In this paper we describe the application of the framework to the parallelization of an AMR numerical solution of low Mach number reacting flows with complex chemistry. developed by Day and Bell [10] extending work on the modeling of incompressible fluid flows by Almgren et al.[1]. Rendleman et al. [15] have described a general framework for the implementation of parallel AMR algorithms, and demonstrated its application to the parallelization of AMR to hyperbolic conservation laws. That framework was developed based on experience gained from researchers in our group and others [8,9,3,7,11]. In that approach, data distribution and communication are hidden in C++ class libraries that isolate the application developer from the details of the parallel implementation.

In the next section we will briefly review the basic algorithmic structure of AMR with emphasis on the particular case of incompressible fluid flow. The dynamic character of AMR leads to a dynamic and heterogeneous work load. In section 3 we discuss the basic parallelization strategy and the load-balancing techniques we use for AMR algorithms. Section 4 provides a description of the parallel implementation (described more fully elsewhere [15]), focusing primarily on the additional methods used to parallelize algorithms specific to AMR for incompressible flows. Finally, we illustrate the use of the program with a simulation of a three dimensional premixed combustion flow.

2 The Adaptive Mesh Refinement Algorithm

AMR solves partial differential equations using a hierarchy of grids of differing resolution. The grid hierarchy is composed of different levels of refinement ranging from coarsest ($l = 0$) to finest ($l = l_{max}$). Each level is represented as the union of non-intersecting logically rectangular grid patches of a given resolution. In this work, we assume the level 0 grid is a grid patch decomposition of a single rectangular parallelepiped, the *problem domain*. In this implementation, the refinement ratio is always even i.e., $\Delta x^{l+1} = \Delta x^l / r$, where r is the refinement ratio, which in the implementation, can be a function of level. The grids are *properly nested*, in the sense that the union of grids at level $l + 1$ is contained in the union of grids at level l for $0 \leq l < l_{max}$, and the level l grids are large enough to guarantee that there is a border at least one level l cell wide surrounding each level $l + 1$ grid. Proper nesting does not require that a fine sub-grid not

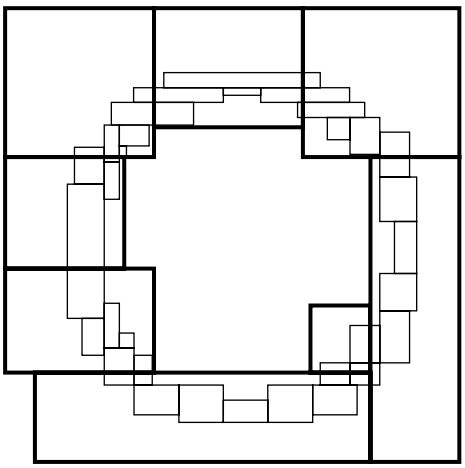


Fig. 1. Two levels of refined grids. Grids are properly nested, but may have more that one parent grid. The thick lines represent grids at the coarse level; the thin lines, grids at the fine level.

cross a coarser sub-grid boundary. Grids at all levels are allowed to extend to the physical boundaries so the proper nesting is not strict there. This is illustrated in Figure 1 in two dimensions. (This set of grids was created for a problem with initial conditions specifying a circular discontinuity.)

Both the initial creation of the grid hierarchy and the subsequent regriding operations in which the grids are dynamically changed to reflect changing flow conditions use the same procedures to create new grids. Cells requiring additional refinement are identified and tagged using an error estimation criteria, possibly using Richardson extrapolation [4]. The tagged cells are clustered [6] into rectangular patches, which in general contain cells that were not tagged for refinement; typically 70% of the cells in a new grid have been tagged by the error estimation process. When new grids are created at level $l+1$, the data on these new grids are copied from the previous grids at level $l+1$ where possible, otherwise the data is interpolated in space from the underlying level l grids.

The AMR algorithm is a recursive procedure that advances each level l, $0 \leq l \leq l_{max}$, with a time-step appropriate to that level, based on CFL considerations. The adaptive algorithm advances the grids at each level independent of other levels in the hierarchy except for obtaining boundary data and the synchronization between levels. The coarser grids supply boundary data in order to integrate finer grids, by filling *ghost cells* in a band around the fine grid data whose width is determined by the stencil of the finite difference scheme. If this data is available from grids at the same level of refinement the data is provided by a simple copy, otherwise it is obtained by interpolation of coarser grid data in time and space. When the coarse and fine grids have been advanced to the same time and we synchronize, there are three corrections that we need to make. First, we replace the coarse data by the volume weighted average of covering fine

grid data. Second, we must correct the coarse cell values by adding the difference between the coarse and fine grid fluxes used to advance grids at their respective levels. Third, we also impose the divergence constraint to the velocity field over the composite grid system. See Almgren et al. [1] for more details on the synchronizations of data between levels.

3 Parallelization of AMR

We have adopted a coarse-grained, message-passing model, using MPI [17,12], in our approach parallelization. This approach is generally associated with distributed memory MIMD architectures, but it can be used on shared memory architectures as well. We make this choice because it enhances portability on distributed memory architectures, and because we feel message-passing programs are more robust. In message-passing parallel programs, the only communication between processors is through the exchange of messages. Direct access to another processor's memory is not provided. In this approach, it is critical to choose carefully which processor has which piece of data. As is apparent from Figure 1, grids vary considerably in size and shape. In AMR the number of grids also changes and is seldom an integer multiple of the number of processors. It is therefore inefficient to assign the grids sequentially to the processors, since the result is unlikely to be load balanced. We will use a load-balancing strategy based on the approach developed by Crutchfield [8,15]. Because most of the data and computational effort is required by the finest level of grids, we need only be concerned with load-balancing the grids on the finest level. In general, the effort required by the coarser grids will be a minor perturbation.

We accept the set of grids provided by the regriding algorithm and seek to find a well-balanced assignment of grids to processors. It turns out to be possible to find well-balanced assignments if we can make the following assumptions.

1. The computational cost of a grid can be estimated using some type of work estimate.
2. The total computational cost of the algorithm is well approximated as the sum of the costs of time-stepping the grids on the finest level. Other costs such as communications, time-stepping coarser grids, regriding, refluxing, etc., are treated as ignorable perturbations.
3. The grids can be approximated as having a broad random distribution, i.e., that the standard deviation of the distribution is not small compared to the average.
4. The average number of grids per processor is at least three.

We have developed an algorithm [8,15] based on an application of the well-known knapsack dynamic programming algorithm, a description of which may be found in Sedgewick's book on algorithms [16]. This approach finds the distribution of grid blocks among processors that results in the smallest total computation time.

While the cost of communication is generally important in message-passing parallel programs—and effort is devoted to its reduction—such costs are ignored

in this load balance scheme. No effort is made to reduce communication costs by placing adjacent grids on the same processor, or on adjacent processors. Since the ratio of communication cost to calculation cost for modern multiprocessors is not overly large, it is reasonable to ignore communication costs in the load balance only if significant computation is done relative to communications. In general, this assumption holds for the application described in this paper because of the amount of work in the chemical reactions calculation.

4 Implementation

The methodology described in this paper has been embodied in a software system that allows for a broad range of physics applications. It is implemented in a hybrid C++/`FORTRAN` programming environment where memory management and control flow are expressed in the C++ portions of the program and the numerically intensive portions of the computation are handled in `FORTRAN`. The software is written using a layered approach, with a foundation library, BoxLib, that is responsible for the basic algorithm domain abstractions at the bottom, and a framework library, AMRLib, that marshals the components of the AMR algorithm, at the top. Support libraries built on BoxLib are used as necessary to implement application components such as interpolation of data between levels, the coarse-fine interface synchronization routines, and linear solvers used in the projection.

The fundamental parallel abstraction is the `MultiFab`, which encapsulates the `FORTRAN` compatible data defined on unions of `Box`s; a `MultiFab` can used as if it were an array of `FORTRAN` compatible grids. The grids that make up the `MultiFab` are distributed among the processors, with the implementation assigning grids to processors using the distribution given by the load balance scheme described in section 3. Non-`MultiFab` operations and data-structures are replicated across all of the processors. This non-parallel work is usually measured to be small. Because each processor possesses the global data layout, the processor can post data send and receive request without a prior query for data size and location.

`MultiFab` operations are performed in one of three ways depending on the implicit communications pattern. In the simplest case, there is no interprocessor communication; the calculation is parallelized trivially with an *owner computes* rule with each processor operating independently on its local data. This is the case with chemistry state evaluations. Different parallel constructs are necessary when data communication involves more than one `MultiFab`, an example of which is the *fill patch* operation, which interpolates from coarse cell data on to overlying fine grid patches. Such constructs cannot be implemented by simply nesting loops because outer loop bodies for sub-grids that are off-processor will not be executed. They must be implemented by our second method using two stages: data is exchanged between processors and then the local targets are updated.

The third, more complicated case, arises in parallelizing loops in the multi-level projection method. The difficulties arise from several causes. First, the

```
WeakSeqCopy(MultiFab& mf_to, const MultiFab& mf_from)
{
  task_list tl;
  for(int i = 0; i < mf_to.length(); ++i)
  {
    for(int j = 0; j < mf_from.length(); ++j)
    {
      tl.add_task(new task_copy(tl, mf_to, i, mf_from, j));
    }
  }
  tl.execute();
}
```

Fig. 2. A weakly sequential implementation of a fill boundary operation

original projection method was implemented using libraries that differ somewhat from the BoxLib/AMRLib libraries, though they share a number of features. For example, the projection uses fill patch operations with different treatments of physical and interior boundary conditions. The major difficulty, however, is a result of the special requirements of the adaptive projection itself. The formulation of the projection algorithm requires that loops be applied in a specific order with possible coupling from loop body to loop body. For example, stencils are evaluated by looping over faces, edges and then corners of grids in the associated `MultiFab`, where results of earlier iterations of the loops may affect the results of subsequent iterates. In addition, the boundary patch filling operation copies in stages, with data from initial stages contributing to data at later stages. The same output patch in an operation may be repeatedly updated in an order specific fashion, and a source patch may require to be updated before being used by an output patch. These order dependencies in operator evaluation give rise to what may be called *weakly sequential loops*.

Weakly sequential loops can be characterized using the language of graph theory. Loop bodies correspond to nodes in the graph and dependencies in the iterations of the loop body correspond to directed edges in the graph. The nodes in the graph can have more than one prior node, and can themselves be the prior nodes of more than one subsequent node. Analogously, weakly sequential loops are similar to the model used in the standard Unix utility, *make*, that manages compilation dependencies. Here a node, usually representing a file, is said to depend on other nodes (files), and can itself be a dependency of other files. The method we use to evaluate weakly dependent loops is again analogous to way the *make* program operates: the *make* program traverses the dependency graph defined in the `Makefile` in such a way that a file is processed by a rule if its dependencies are up to date.

We will use the example in Figure 2 to illustrate the use of weakly sequential loops. The `add_task` method of the `task_list`, is responsible for inserting a task into the task loop and evaluating dependencies of the current task on prior tasks

```
Procedure task_list::execute()
    while task list not empty
        Pop head of task list into task T.
        if T has no outstanding dependencies
            if T has not been started, Start T, post message passing requests
            if T is message complete, Execute T and mark as finished;
        else
            Push T at end of task list
        endif
    end while
End Procedure task_list::execute
```

Fig. 3. Algorithm for `task_list::execute`

in the loop. The `task_copy` is a helper-class that implements copying of data from sub-grid j of the `MultiFab` mf_from to sub-grid i of mf_to. The `task_list` is passed to the constructor of the `task_fill` operation so that dependencies can be detected among loop iterations. This style of coding has the effect of flattening multiply nested parallel loops into a single serial loop that is processed as indicated below.

The `execute` member of `task_list` used in Figure 2 causes each task to be executed in `task_list` using the algorithm in Figure 3. The loop attempts to maximize concurrency by using asynchronous message passing calls. Potentially, many messages requests can be outstanding, though in practice MPI implementations restrict the number of outstanding posted messages. For that reason, the loop is "throttled" by limiting the number of active members of the task loop. When a task is marked as finished, it is cleared from the dependency list of all remaining tasks in the loop.

Naive implementation of the construction of the dependency graph results in an operation count of $O(N^2)$, where N is the number of loop elements, usually proportional to the number of grids at a level. This could be significant for the case when there are thousands of elements in the loop, a situation that is not uncommon. However, careful implementation reduces computation cost to $O(N/P)^2$, which exhibits lower growth because the number of processors, P, used in a calculation is an increasing function of the number of grids. The careful implementation removes tasks from the task loop as they are added if the task does not use data local to that processor, or if it uses data only local to that processor and the task does not depend on prior tasks in the task list.

The principle disadvantage of the task list approach is that it encourages an unnatural coding style: a helper class must be implemented for each loop in the program. For the projection, which consists of approximately 13,000 lines of C++ (and 16,000 lines of `FORTRAN`), less than a dozen helper classes are needed.

5 Numerical Example

The example we choose to illustrate the use of the program represents a simplified model of spray fueling in an internal combustion engine. A spray of premixed fuel droplets enters a turbulent combustion chamber, where it is heated, evaporated and burned. Here, the fuel-rich spray is assumed to be heated to 1000 K by other processes, and the calculation evolves the combustion of fuel in two simultaneous combustion modes: a fast premixed burn of the heated fuel, and a slower diffusion flame at the interface between the air and remaining fuel. In this model, a 1 cm radius sphere of 1000 K hydrogen-air mixture (equivalence ratio = 4) is used to represent the fuel. A domain of 10x10x10 cm, co-centered with the initial fuel sphere, is initialized with room temperature isotropically turbulent air. All the boundaries of the domain are outflowing. In the initial stages, the fuel sphere expands from the temperature rise due to the fast premixed burn. The expansion, and resulting interaction with the background turbulence, generates surface instabilities at the interface between the fuel and air, increasing the interface area, reactant mixing, and overall consumption rate of the fuel. 9 chemistry species (H_2, H, O, O_2, OH, H_2O, HO_2, H_2O_2, and N_2) with 27 reactions among them are used to model the combustion process.

The simulation was performed on the IBM Power3 SMP system at the U.S. Army Engineer Research and Development Center, Major Shared Resource Center in Vicksburg, MS. The AMR parameters used specified 2 levels of refinement with a refinement ratio of 2. The concentration of species H_2O_2 is used to mark the zones in the model that require refinement. The coarse grid consisted of 32 zones in each coordinate direction. With these refinement ratios the finest level has an effective resolution of 128 zones in each coordinate direction. For this relatively small model, only 8 processors were used, in accord with our load-balance guidelines of assigning 3 or more grids per processor (section 3). Figure 4 show a rendering of the temperature at time step 380. At this point in the simulation, there are 34 grids at the finest level, covering approximately 18% of the domain. Approximately 30% of the computational time is consumed in evaluating the chemical reaction processes. The remainder of the time is roughly allocated to 5% for scalar advection, 20% for the velocity projection, and 30% for diffusion of the chemical species. The remainder of the time is charged to overhead associated with the adaptive algorithm and with the technique, not described here, used to load balance the chemistry [14].

6 Conclusions

We have described the techniques used to parallelize an AMR variable density, viscous, incompressible flow solver targeted to low-Mach number reacting flows. One of the applications of this program, together with its companion program for hyperbolic systems of conservation laws described in Rendleman et al. [15], is to provide end-to-end simulation capabilities for explosions in buried chamber systems. The hyperbolic code is used to examine the prompt effects of the explosion,

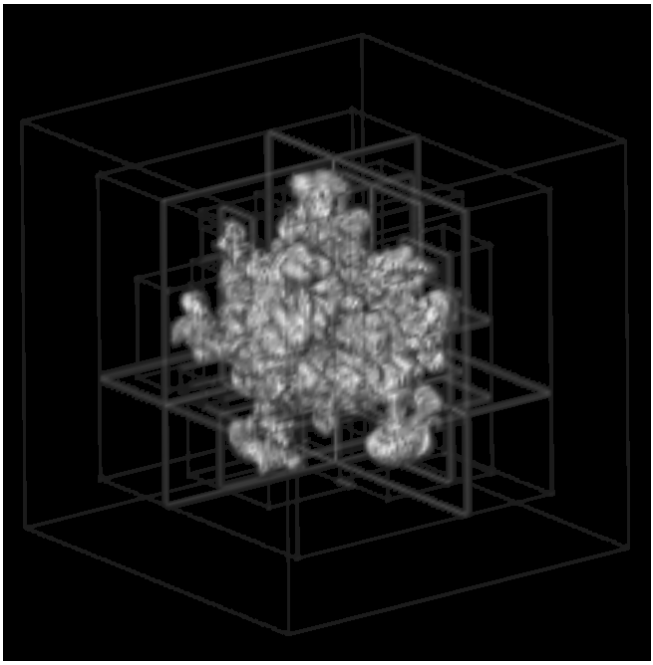


Fig. 4. High temperature at fuel-air interface rendered at time step 380. Solid lines indicate refined zones at the finest two resolutions. The coarse grids is not shown.

while the low-Mach number code is used to monitor longer time scale processes associated with burning of the chamber system's contents. The methods used are rectangular sub-grid decomposition of data among parallel processors and a use of SPMD style programming constructs. Load balance is achieved using an efficient and effective dynamic-programming algorithm. We also described our software methodology including a novel technique for identifying and evaluating weakly sequential loops. We demonstrated the use of the program for an example of pre-mixed fuel air combustion in a room temperature isotropically turbulent body of air.

Acknowledgements

The authors wish to thank J. B. Bell and W. Y. Crutchfield for valuable input. We also wish to thank M. S. Day for providing the numerical example used in section 5.

References

[1] A. S. Almgren, J. B. Bell, P. Colella, L. H. Howell, and M. L. Welcome. A conservative adaptive projection method for the variable density incompressible Navier-Stokes equations. *J. Comput. Phys.*, 142:1–46, May 1998.

[2] J. Bell, M. Berger, J. Saltzman, and M. Welcome. A three-dimensional adaptive mesh refinement for hyperbolic conservation laws. *SIAM J. Sci. Statist. Comput.*, 15(1):127–138, 1994.

[3] M. Berger and J. Saltzman. AMR on the CM-2. *Applied Numerical Mathematics*, 14:239–253, 1994.

[4] M. J. Berger and P. Colella. Local adaptive mesh refinement for shock hydrodynamics. *J. Comput. Phys.*, 82(1):64–84, May 1989.

[5] M. J. Berger and J. Oliger. Adaptive mesh refinement for hyperbolic partial differential equations. *J. Comput. Phys.*, 53:484–512, March 1984.

[6] M. J. Berger and J. Rigoutsos. An algorithm for point clustering and grid generation. *IEEE Transactions on Systems, Man, and Cybernetics*, 21:1278–1286, 1991.

[7] P. Colella and W. Y. Crutchfield. A parallel adaptive mesh refinement algorithm on the C-90. In *Proceedings of the Energy Research Power Users Symposium*, July 11–12, 1994.
`http://www.nersc.gov/aboutnersc/ERSUG/meeting_info/ERPUS/colella.ps`.

[8] W. Y. Crutchfield. Load balancing irregular algorithms. Technical Report UCRL-JC-107679, Applied Mathematics Group, Computing & Mathematics Research Division, Lawrence Livermore National Laboratory, July 1991.

[9] W. Y. Crutchfield. Parallel adaptive mesh refinement: An example of parallel data encapsulation. Technical Report UCRL-JC-107680, Applied Mathematics Group, Computing & Mathematics Research Division, Lawrence Livermore National Laboratory, July 1991.

[10] M. S. Day and J. B. Bell. Numerical simulation of laminar reacting flows with complex chemistry. *Combust. Theory Modelling*, 4:535–556, 2000.

[11] J. A. Greenough, W. Y. Crutchfield, and C. A. Rendleman. Numerical simulation of a wave guide mixing layer on a Cray C-90. In *Proceedings of the Twenty-sixth AIAA Fluid Dynamics Conference*. AIAA-95-2174, June 1995.

[12] W. Gropp, E. Lusk, and A. Skjellum. *Using MPI: Portable Parallel Programming with the Message-Passing Interface*. Scientific and Engineering Computation. The MIT Press, Cambridge, Mass, 1994.

[13] R. B. Pember, L. H. Howell, J. B. Bell, P. Colella, W. Y. Crutchfield, W. A. Fiveland, and J. P. Jessee. An adaptive projection method for unsteady low-Mach number combustion. *Comb. Sci. Tech.*, 140:123–168, 1998.

[14] Charles A. Rendleman, Vincent E. Beckner, and Marc S. Day Mike Lijewski. Parallel performance of an adaptive mesh refinement method for low mach number combustion. in preparation, January 2001.

[15] Charles A. Rendleman, Vincent E. Beckner, Mike Lijewski, William Crutchfield, and John B. Bell. Parallelization of structured, hierarchical adaptive mesh refinement algorithms. *Computing and Visualization in Science*, 3(3):147–157, 2000.

[16] R. Sedgewick. *Algorithms in C++*. Addison-Wesley Publishing Company, Reading, Massachusetts, 1992.

[17] M. Snir, S. Otto, S. Huss-Lederman, D. Walker, and J. Dongarra. *MPI: The Complete Reference*. Scientific and Engineering Computation. The Mit Press, Cambridge, Mass, 1996.

Combustion Dynamics of Swirling Turbulent Flames

Suresh Menon, Vaidyanathan Sankaran, and Christopher Stone

School of Aerospace Engineering
Georgia Institute of Technology
Atlanta, Georgia 30332
suresh.menon@aerospace.gatech.edu

Abstract. A generalized Large-Eddy Simulation (LES) methodology has been developed to simulate premixed and non-premixed gas-phase and two-phase combustion in complex flows such as those typically encountered in gas-turbine combustors. This formulation allows the study and analysis of the fundamental physics involved in such flows, i.e., vortex/flame interaction, combustion dynamics and stability, fuel-air mixing, droplet vaporization, and other aspects of combustion. Results for swirling premixed combustion undergoing combustion instability and for swirling spray combustion in full-scale gas turbine engines are discussed here. Results show that swirl can stabilize combustion in premixed system and can reduce the magnitude of the amplitude of the pressure oscillation. In two-phase systems, significant modification to the high shear regions due to vaporization of droplets is observed. Droplets are also seen to concentrate in regions of low vorticity and when they vaporize, the gaseous fuel gets entrained into regions of high vorticity. This process plays a major role in fuel-air mixing and combustion processes in two-phase systems.

1 Introduction

The simulation of compressible, swirling, turbulent reacting flows such as those found in contemporary power generation and aircraft gas turbine systems pose a great challenge due to the widely varying time and length scales that must be resolved for accurate prediction. In addition to the difficulty in resolving all the important turbulent scales, the presence of multiple modes/regimes of combustion and the interaction between various chemical species and physical phases (liquid and gas) in the same combustion device further complicates the modeling requirements. Here, a methodology based on Large-Eddy Simulations (LES) is developed and applied to these types of problems.

In LES, scales larger than the grid scale are computed using a time- and space-accurate scheme while the effects of the smaller, unresolved scales (assumed to be mostly isotropic) are modeled. For momentum transport closure simple eddy viscosity based sub-grid models are sufficient since the unresolved small-scales primarily provide dissipation for the energy transferred from the

V.N. Alexandrov et al. (Eds.): ICCS 2001, LNCS 2073, pp. 1127–1136, 2001.
© Springer-Verlag Berlin Heidelberg 2001

large scales. However, combustion occurs at the molecular scales and the interaction between the small-scales of motion and molecular diffusion play a major role in combustion and heat release. Thus, to properly account for heat release effects, the small scale processes must be simulated accurately (which is in conflict with the eddy viscosity approach used for momentum closure). In order to deal with these distinctly different modeling requirements, a sub-grid combustion model has been developed [1] that resides within each LES cell and accounts for the interaction between the small-scale mixing and reaction-diffusion processes.

Earlier studies [1,2,3,4] have established the ability of the LES model in premixed and non-premixed systems. In the present study, the dynamics of swirling premixed flames in a gas turbine combustor is studied using the same LES approach. For two-phase reacting flow, this LES approach was extended [5] and used within a zero-Mach number (incompressible) formulation to study spray transport and vaporization in spatial mixing layers. In the present study, the two-phase model has been implemented within the compressible LES model developed earlier [2,4] for gas phase combustion and then used to study spray combustion in a high Reynolds number swirl flow in a gas turbine combustor.

2 Large-Eddy Simulation Model

The Favre filtered mass, momentum, energy, and species conservation equations are solved in the LES approach. In addition, the unresolved turbulence kinetic energy is modeled with a single sub-grid kinetic energy equation, k^{sgs} [2]. The sub-grid kinetic energy is used to close the unresolved stresses and energy/species flux terms resulting from the filtering operations. For premixed combustion studies, a thin-flame model [2] is employed while a Lagrangian droplet tracking method [6] is used to explicitly track the droplets in the Eulerian gas field in the spray simulations. In this method, the liquid droplets are tracked with a Lagrangian approach to explicitly compute the mass, momentum, energy and species transfer between the continuum and dispersed phase. The gas phase LES velocity fields and the sub-grid kinetic energy are used to estimate the instantaneous gas velocity at the droplet location. Drag effects due to the droplets on the gas phase is explicitly included. Heat transfer from gas phase to the liquid phase aids in the vaporization and the subsequent mass transfer to the gas phase. This provides the thermal coupling between the two phase. Thus, full coupling is achieved between the two phases in the simulation.

The governing equations mentioned above have been withheld for brevity; however, they along with further details can be found elsewhere [2,5].

3 Numerical Methodology

The LES equations of motion are solved on a three dimensional, boundary-conforming, grid using an explicit finite-volume scheme that is fourth-order accurate in space and second-order accurate in time. No-slip, adiabatic wall con-

ditions conditions are used along with non-reflecting inflow/out-flow boundary conditions [7].

The configuration used for both premixed and two-phase combustion studies consists of an inlet pipe expanding into the larger cylindrical combustion chamber. A swirling velocity profile with a swirl number of 0.56 is imposed at the inlet boundary. The mean inlet mass flow rate, temperature, and pressure are 0.435 Kilograms/second, 673 Kelvin, and 11.8 atmospheres, respectively. The Reynolds number based on inlet bulk velocity and inlet diameter is 330,000. An inflow turbulent field is generated by using a specified turbulence intensity (7%) on a randomly generated Gaussian field.

For two-phase (spray) LES, a dilute spray is introduced at the inlet plane using 20 μm droplets (future studies will incorporate a log-normal size distribution). The Stokes number, the ratio of droplet to flow time scales, is approximately 8.2. Droplets below a cut-off size of 5 μm are assumed to instantly vaporize and mix. Gas phase velocities at the particle locations are interpolated using a fourth-order scheme. The governing Lagrangian (two-phase) equations are integrated with a fourth-order Runge-Katta scheme. Elastic collisions are assumed for particle/wall interaction.

A grid resolution of $141 \times 65 \times 81$ is employed for both the premixed and two-phase LES. Clustering of the grid in regions of high shear is used. For the spray simulations, 120,000 droplet groups are tracked in the computational domain.

The LES solver is implemented on massively parallel systems using domain decomposition and standard Message-Passing Interface (MPI) libraries. The parallel algorithm exhibits good scalability (85% parallel efficiency on 128 CPU's) on several high-performance computing platforms. Simulations on the Cray T3E-900 typically require 900 and 3400 CPU hours for a single flow-through (the time for a fluid element to traverse the entire computational domain) for the premixed and spray calculations, respectively. In general, 5 to 10 flow-through-times are simulated for statistical analysis. The memory requirements for the premixed and spray computations are 2.9 and 12.3 Gigabytes, respectively.

4 Combustion Dynamics in a Premixed System

Accurate prediction of the coupling between unsteady heat release and pressure oscillation is critical to simulate combustion dynamics in dump combustors. However, proper resolution of this coupling is especially difficult due to the unsteadiness of the controlling processes (e.g., fuel injection) and the high nonlinearity of the interactions between turbulent mixing, acoustic wave motion, and unsteady heat release. Large-scale structures in the flow play a key role in the coupling process by controlling the fuel-air mixing. In non-swirling flows, axisymmetric coherent vortices are shed at the dump plane and these structures interact with the acoustic waves and heat release in the combustor. In a highly swirling flow, azimuthal (helical) instability modes are present and the interaction between the modes of motion is more complicated. In fact, swirl can stabilize or even destabilize instability in sudden expansion flows [8]. Therefore, the identification

of flow or system parameters that control swirl induced instability is extremely important for the design of stable combustion systems.

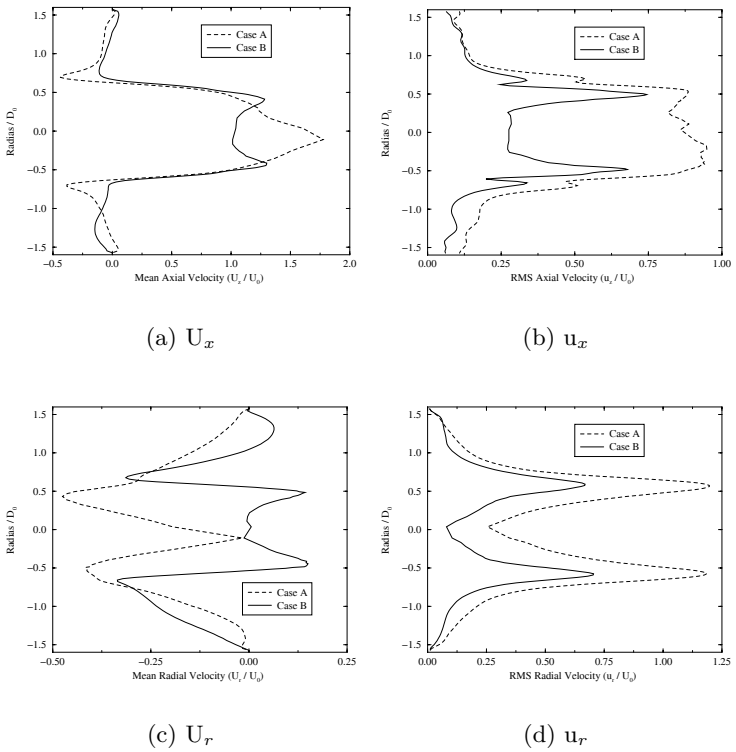
The effect of swirl on lean premixed flames is investigated in this study. Although lean burning systems have some significant advantages (such as reduced pollutant emission and increased fuel efficiency), lean operation is sensitive to small fluctuations which under certain conditions can get amplified resulting in high-amplitude pressure oscillations. This phenomenon is often called combustion instability and understanding this phenomenon is the focus of the current study.

Two simulations are conducted in order to observe the effects of heat release on the dynamics of combustion in the swirling combustor. Case A simulates a passive flame (zero heat-release) while case B includes heat-release with a flame temperature of 1807 Kelvin. In the limit of zero heat release, the thin-flame field does not effect the flow and therefore, acts as a passive scalar that is advected by the fluid flow without affecting the flow field. In the heat release case, the flame responds to the flow field that is modified due to heat release and thermal expansion effects.

The mean and fluctuating axial and radial velocity profiles across the diameter of the combustor (at the center plane) are shown in Fig. 1. The profiles are shown at a non-dimensional distance of $\mathrm{X/D_0} = 0.2$ (D_0 is the diameter of the inlet pipe) downstream of the dump plane. Near the centerline, the mean axial velocity seems to be reduced due to heat release while the radial velocity is increased. Assuming a conical flame, the flame-normal expansion will be predominantly aligned in the radial direction. This divergence (broadening of the streamlines) would cause the reduction in the mean axial velocity inside the flame region and the corresponding increase in the radial component. Additionally, the magnitude of recirculation after the sudden expansion is reduced in the presence of heat release.

Root-Mean-Square (RMS) velocity profiles at the same axial location are shown in Figs. 1(b, d). In the shear layer region, $\mathrm{R/D_0} \approx 0.5$, both simulations show high fluctuations; however, the inner core region has some distinct differences. Similar to its mean counterpart, the fluctuating axial velocity is reduced near the centerline. The other two components show the same trend of lower velocity fluctuations associated with heat release. This reduction is caused by the increased viscous dissipation in the hot products.

Shown in Fig. 2 are the pressure fluctuation spectra for the two cases. The pressure signals were recorded at the base of the dump plane where the vorticity is low. Both simulations reveal a dominant frequency (plus a harmonic) at a Strouhal number (defined as $f\,\mathrm{D_0/U_0}$) of 0.88 for the cold flow and 1.12 for the reacting flow (with harmonics at 1.76 and 2.24, respectively). Analysis of the pressure amplitudes and phase angles of these pressure signals along the longitudinal axis of the combustor indicated a 3/4 wave with a wave length proportional to the combustor length (from dump plane to diffuser). This wave shape is acoustic as indicated by the frequency shift from a cold flow (Case A)



(a) U_x (b) u_x

(c) U_r (d) u_r

Fig. 1. Mean and RMS velocity profiles at $X/D_0 = 0.2$ downstream of the dump plane. (a, b): Axial (U_x, u_x) and (c, d) radial (U_r, u_r).

to the hot (Case B) (i.e., the frequency shift corresponds to the change in the speed of sound at the higher temperature).

As with the RMS velocity profiles, Fig. 2(a) gives credence to the attenuation effect of heat release. While the acoustic wave shape is the same, the amplitude is attenuated by almost 700%. A time segment of the global Rayleigh parameter ($\bar{R}(t)$) [9] (not to scale) is given in Fig. 2(b). Positive $\bar{R}(t)$ corresponds to amplification while negative indicates neutral oscillations or attenuation. This time sequence shows stable operation ($\bar{R}(t)$ is nearly always (-)), i.e. the pressure, p', and heat release, $\Delta q'$, fluctuations are not in phase. Only at the higher harmonic of the pressure signal (St = 2.24) does amplification occur, indicating that the heat release oscillations are at or near the higher frequency.

Figure 3 shows a mean and an instantaneous view of the three-dimensional flame surface. All pictures were taken with the same frame of references to allow direct comparison of the flame dimensions. Due to heat release and the associated thermal expansion, the mean flame surface is repelled (expands) outward and is

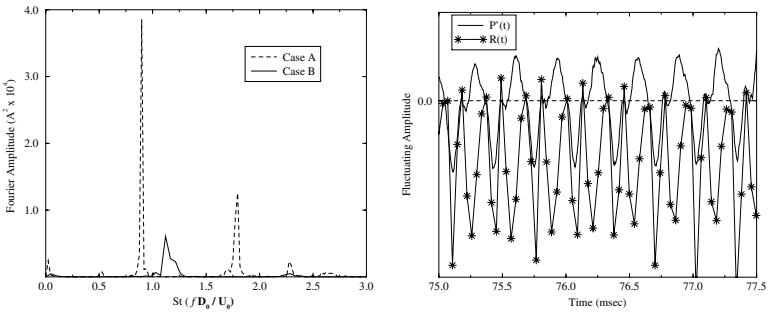


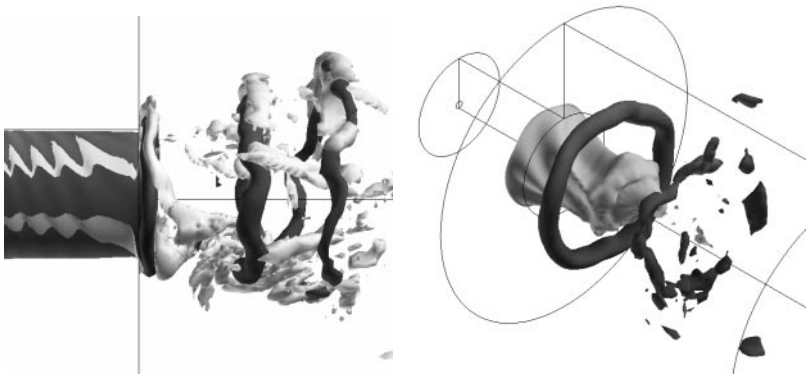
Fig. 2. (a) Fourier transform of pressure fluctuations in the combustor. (b) Time sequence of the volume averaged Rayleigh parameter, $R(t)$.

(a) (b) (c) (d)

Fig. 3. Mean and instantaneous 3D flame surface: Case A (a) mean and (b) instantaneous, and Case B (c) mean and (d) instantaneous. Flow direction direction is from top left to bottom right.

also longer. However, no significant visual distinction can be made between the cold and hot instantaneous flame fronts. Both show wrinkling/elongation in the azimuthal direction (rib-shaped structures).

The rib-shaped structures in the flame front are aligned with vortex tubes generated in the swirling boundary layer that is shed from at the dump plane. An example of the vortex shedding from Case B is shown in Fig. 4(a) & (b). Fig. 4(a) shows the azimuthal (ω_θ, dark gray) and axial (ω_x, light gray) vorticity. The large-scale, ring structures are predominantly ω_θ. Braid structures (mostly ω_x) are observed in the region between the shed vortices. As the ring vortices shed, they entrain the flame and carry it along (shown in Fig. 4(b)). The flame is drawn outwards till the vortex breaks down. This vortex-flame interaction forms a flame oscillation cycle with a time-scale proportional to the vortex shedding rate. The shedding rate is strongly coupled with the longitudinal acoustic waves



(a) (b)

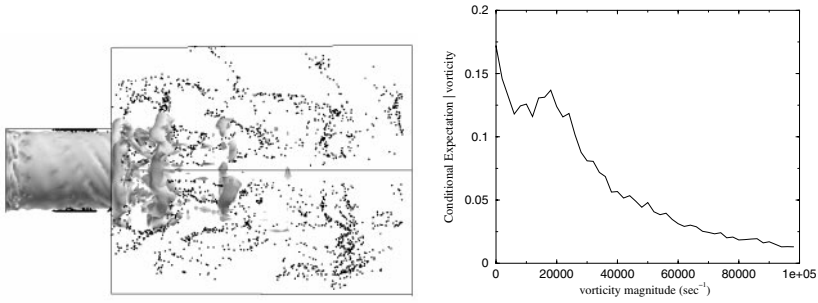
Fig. 4. Vortex-Flame interactions in the combustor (instantaneous views): (a) Tangential Vorticity, ω_θ (dark gray) and Axial Vorticity ω_x (light gray), (b) Flame surface, G, (light gray) and Tangential Vorticity, ω_θ (dark gray). Flow direction is from top left to bottom right in (b).

in the combustor (left running waves trip the unstable boundary layer at the dump plane causing vortex roll-up).

5 Spray Combustion

Three simulations are discussed here. A non-reacting gas-phase-only case, a non-reacting two-phase case (i.e., only momentum coupling) and a reacting (using a global infinite-rate kinetics) two-phase case. In all three cases, the same swirling inflow was employed. A general observation, for all three cases, shows that the heavy particles do not follow the gas phase due to their larger inertia; however, as they get smaller, due to vaporization, they equilibrate with the gas phase. Smaller droplets are observed in the recirculation bubble near the dump plane. On the other hand, in the momentum coupled case, fewer particles are seen in the recirculation bubble due to the large Stokes number associated with the particles. Large dispersion of droplets towards the outer region of the combustor (not shown) is observed. This radial spread is seen to increase with downstream distance. Larger particles that reach the wall bounce back and move downstream. Further downstream, the distribution of the droplets tends to be more uniform.

Figure 5(a) shows the ISO-surface of the vorticity and droplet distribution in the combustor. Coherent vortex structures which are seen near the dump plane in the gas-phase simulations are quickly disintegrated in the presence of droplets. Analysis shows that droplets tend to accumulate in regions of low vorticity. This



(a) (b)

Fig. 5. (a) Contours of Vorticity ISO-surface (45,000 s^{-1} (yellow) and the droplet number distribution and (b) Conditional expectation of droplet number density.

type of preferential accumulation of droplets in regions of low vorticity was also observed in earlier studies of simpler shear flows [5]. Conditional expectation of droplet number density conditioned on vorticity is shown in Fig. 5(b). The abscissa in this plot is the normalized by the vorticity magnitude. As can be seen, the probability density function is asymmetric and is biased towards low vorticity.

Figures 6(a) and (c) show respectively, the mean gas-phase velocity profiles in the stream-wise and transverse directions. These radial profiles are shown at a non-dimensional distance of 0.14 (which is slightly downstream of the dump plane). As was observed in the premixed case, the mean axial velocity is reduced with heat release. In addition, the presence of particles (with or without heat release) reduces the mean velocity, especially in the shear layer region. Thus, particle drag effects reduces large radial variation in the velocity profiles while heat release (and the associated thermal expansion) further smooths out radial gradients.

Further analysis shows that the swirl has been significantly attenuated due to the presence of the droplets. It should be noted that for flows with a large droplet to gas-phase density ratio and droplet sizes smaller than the Kolmogorov scale, the particle paths, the relative velocities (between the two phases), and the particle drag are all uniquely determined by the Stokes number. Therefore, future studies at different Stokes numbers and more realistic droplet size distributions are needed.

Figures 6(b) and (d) show respectively, the root mean square velocity fluctuations in the stream-wise and transverse directions. It can be seen that the turbulent fluctuations have been attenuated in the presence of the particles. The presence of droplets decreases the turbulence level by introducing additional dissipation. In particular, turbulent fluctuations have been attenuated significantly

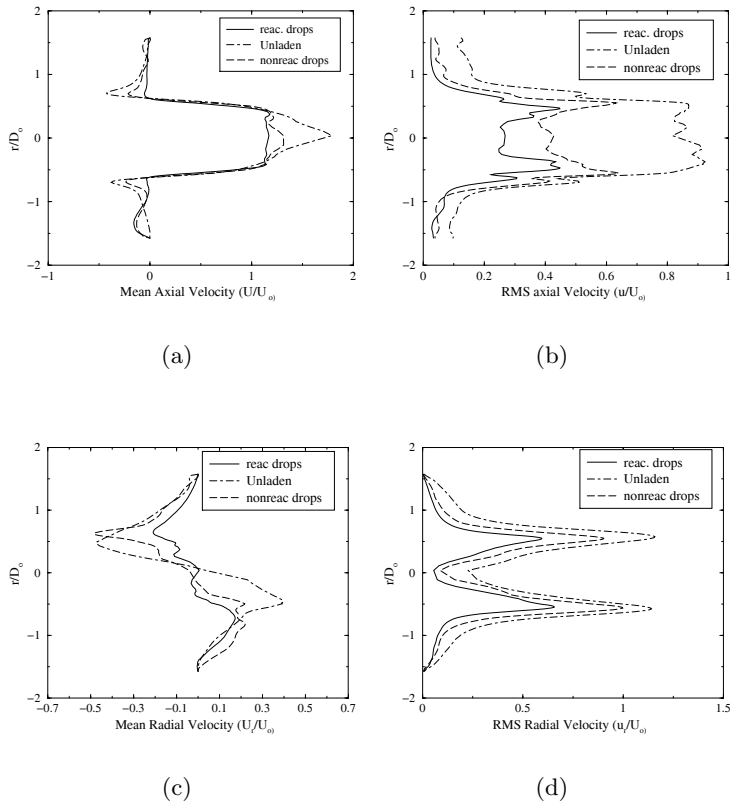


Fig. 6. Mean and RMS velocity profiles at $X/D_0 = 0.14$ downstream of the dump plane. (a, b): Axial (U_x, u_x) and (c, d) radial (U_r, u_r).

in regions where turbulent intensities are high in unladen flows (i.e., in regions of high shear). This is because in these regions, the local Stokes number based on the turbulent time scales is high leading to increased attenuation of the turbulence closer to the shear layer where the turbulence production is very high. This observation is consistent with an earlier study [10]. Turbulence levels in the recirculating zones are not affected significantly, due to the presence of fewer particles there.

6 Conclusions

Simulation of high Re, swirling premixed and spray flames in full-scale gas turbine combustors have been carried out using the same LES solver. This LES approach includes a more fundamental treatment of the interaction between the

flame, gas-phase, and liquid flow dynamics. Combustion dynamics in the lean premixed system has been simulated and results show that the dominant mode shape is the three-quarter acoustic wave shape in the combustor. Results also show that swirl and heat release effects can stabilize the system by reducing the amplitude of pressure oscillation. Simulation of spray combustion show that many global features such as the preferential concentration of droplets in low vorticity regions, droplet dispersion and turbulence modification by the particles are all captured reasonably well. However, many other issues such as the effect of the mass loading ratio, droplet vaporization rate, and Stokes number on the turbulent reacting flow needs to be studied further. These issues are currently being addressed and will be reported in the near future.

7 Acknowledgments

This work was supported in part by the Army Research Office (ARO) under the Multidisciplinary University Research Initiative (MURI) and General Electric Power Systems. Computational time was provided by DOD High Performance Computing Centers at NAVO (MS), SMDC (AL), WPAFB (OH), and ERDC (MS) under ARO and WPAFB HPC Grand Challenge Projects.

References

1. V.K. Chakravarthy and S. Menon, "Large-eddy simulations of turbulent premixed flames in the flamelet regime," *Combustion Science and Technology*, vol. 162, pp. 1–48, 2001, to appear.
2. W.-W. Kim, S. Menon, and H. C. Mongia, "Large eddy simulations of a gas turbine combustor flow," *Combustion Science and Technology*, vol. 143, pp. 25–62, 1999.
3. W.-W. Kim and S. Menon, "Numerical modeling of fuel/air mixing in a dry low-emission premixer," in *Recent Advances in DNS and LES*, Doyle Knight and Leonidas Sakell, Eds. Kluwer Academic Press, 1999.
4. W.-W. Kim and S. Menon, "Numerical modeling of turbulent premixed flames in the thin-reaction-zones regime," *Combustion Science and Technology*, vol. 160, pp. 110–150, 2000.
5. S. Pannala and S. Menon, "Large eddy simulations of two-phase turbulent flows," *AIAA 98-0163, 36th AIAA Aerospace Sciences Meeting*, 1998.
6. J. C. Oefelein and V. Yang, "Analysis of transcritical spray phenomena in turbulent mixing layers," *AIAA 96-0085, 34th AIAA Aerospace Sciences Meeting*, 1996.
7. T.J. Poinsot and S.K. Lele, "Boundary conditions for direct simulations of compressible viscous flow," *Journal of Computational Physics*, vol. 101, pp. 104–129, 1992.
8. S. Sivasegaram and J.H. Whitelaw, "The influence of swirl on oscillations in ducted premixed flames.," *Combustion Science and Technology*, vol. 85, 1991.
9. S. Menon, "Active combustion control in a ramjet using large-eddy simulations," *Combustion Science and Technology*, vol. 84, pp. 51–79, 1992.
10. J. R. Fessler and Eaton J. K., "Turbulence modification by particles in a backward facing step flow," *Journal of Fluid Mechanics*, vol. 394, pp. 97–117, 1999.

Parallel CFD Computing Using Shared Memory OpenMP

Hong Hu and Edward L. Turner

Department of Mathematics
Hampton University, Hampton, VA 23668, USA
hong.hu@hamptonu.edu

Abstract. The eXtended Full-Potential (FPX) helicopter rotor Computational Fluid Dynamics (CFD) code in its reduced two-dimensional version is successfully converted into a parallel version. The FPX code solves the full potential equation using an approximately factored finite-difference scheme. The parallel version of the code uses Open Multi-Processing (OpenMP) directives as parallel programming tool. Open MP based parallel code is portable and can be compiled with Fortran compiler that supports the OpenMP Fortran standard. OpenMP based parallel code can also be compiled using non-parallel Fortran compiler into a serial executable. The performance study of the parallel code is made. The results show that OpenMP is easy to use and a very efficient parallel programming tool for the present problem. Keywords: Parallel Computing, Computational Fluid Dynamics

1 Introduction

Computational Fluid Dynamics is the one of the areas that needs super-fast computation power. The numerous calculations that are needed to execute CFD codes may require hours and even days of Central Processing Unit (CPU) time. Parallel computation using more than one CPU is highly considered in the field of Computational Fluid Dynamics. Parallel computation allows CFD codes to run fast, since the computational workload is distributed among computer processors.

There are two major approaches in multiprocessing parallel computational architectures: distributed memory where each CPU has a private memory, and shared memory where all CPUs access common memory. Different parallel processing architectures give the different parallel performance characteristics, and different applications perform differently on different architectures. Today's new multiprocessing parallel computers are a combination of the best parts of shared- and distributed- memory architectures, such as distributed shared- memory system of Silicon Graphics (SGI) Origin 2000. Parallel program can be developed on the SGI Origin 2000 using either a shared- memory or distributed- memory architecture.

Open MP shared-memory parallel processing is employed in the present work to develop a parallel version of a helicopter rotor FPX CFD code in a reduced two-dimensional form. This paper presents the work on the parallel code development along with the performance analysis of the resulting parallel code.

V.N. Alexandrov et al. (Eds.): ICCS 2001, LNCS 2073, pp. 1137–1146, 2001.
© Springer-Verlag Berlin Heidelberg 2001

2 Methodology of the FPX CFD Code

While in the fixed-wing aerodynamic computational community increasingly expensive and complex Euler and Navier-Stokes methods have been used recently, potential methods are still major analysis tools in rotary-wing aerodynamics computational community. The FPX [1] rotor code is an efficient and accurate potential method in this field. The code represents an industry standard for rotary-wing computations. The FPX code is a modified and enhanced version of Full-Potential Rotor (FPR) code [2]. The code (either FPX or FPR) solves three-dimensional unsteady full-potential equation. The code has been used in various helicopter hover and forward flight cases. The application of the code produces excellent results.

The unsteady, three-dimensional full-potential equation in strong conservative form in blade-fixed body-conforming coordinates (ξ, η, ζ, τ) is written as

$$\frac{\partial}{\partial \tau}\left(\frac{\rho}{J}\right)+\frac{\partial}{\partial \xi}\left(\frac{\rho U}{J}\right)+\frac{\partial}{\partial \eta}\left(\frac{\rho V}{J}\right)+\frac{\partial}{\partial \zeta}\left(\frac{\rho W}{J}\right)=0 \tag{1}$$

with

$$\rho=\{1+\tfrac{\lambda-1}{2}[-2\Phi_\tau-(U+\xi_t)\Phi_\xi-(V+\eta_t)\Phi_\eta-(W+\zeta_t)\Phi_\zeta]\}^{\frac{1}{\gamma-1}} \tag{2}$$

where Φ is the velocity potential, U, V and W are contravariant velocity components, ρ is the density, and J is the grid Jacobian.

The FPX/FPR codes solve Eq. (1) using an implicit finite-difference scheme, where the time-derivative is replaced by a first-order backward differencing and the spatial-derivatives are replaced by second-order central differencing. The resulting difference equation is approximately factored into three operators L_ξ, L_η and L_ζ in ξ, η and ζ directions, respectively,

$$L_\xi L_\eta L_\zeta(\Phi^{n+1}-\Phi^n)=RHS \tag{3}$$

The detail of the scheme is presented in [1,2].

The FPX is the substantially modified version of the FPR code. Both entropy and viscosity corrections are included in the FPX code. The entropy correction potential formulation accounts for shock produced entropy to enhance physical modeling capabilities for strong shock cases. Either a two-dimensional or three-dimensional boundary layer model is coupled with the FPX code to account for viscosity effects.

In addition, an axial flow capability is added into the FPX code to treat tilt-rotors in forward flight.

3 Parallel Implementation

3.1 Background

SGI Origin 2000 High Performance Computer (HPC) is chosen as the platform to use for developing the parallel code. The SGI Origin 2000 is a distributed shared-memory system, with hardware designed like distributed-memory architecture. However, the system keeps track of which memory space holds which variable, therefore parallel programs can be developed using either a shared or distributed-memory model on the SGI Origin 2000. The programmer can use either Message Passing Interfaces (MPIs) for distributed memory programming, or OpenMP for shared-memory programming, or some combinations of both to best suit the application [3].

MPI has become accepted as a portable style of distributed-memory parallel programming, but has several significant weaknesses that limit its effectiveness and scalability [4]. Message passing in general is difficult to program and doesn't support incremental parallelization of an existing sequential program. The MPI is therefore not chosen for this work.

Shared-memory parallel programming directives have not been standardized in the industry before the introduction of OpenMP. An earlier standardization effort was never formally adopted. Thus, vendors have each provided a different set of directives, very similar in syntax and semantics, and each used a unique comment or programming notation for "portability". OpenMP consolidates these directive sets into a single syntax and semantics, and finally delivers the long-awaited promise of single source portability for shared-memory parallelism [5]. OpenMP is a specification for a set of compiler directives, library routines, and environment variables for specifying shared memory parallelism. The OpenMP is available for both Fortran and C/C++ languages. The FPX rotary code was written in Fortran, therefore OpenMP for Fortran is used as the parallel-programming tool.

OpenMP directives are portable and can be compiled using non-MIPSpro Fortran Compiler that supports the OpenMP Fortran standard. The parallel code developed on SGI Origin 2000 can be executed on the Sun Supercomputer and IBM Power 3 computer, for examples.

Fig. 1 gives an example of using OpenMP directive, where "C$OMP PARALLEL DO" directive instructs the parallel Fortran compiler to compile the loop into parallel executable. It should be mentioned that every OpenMP directive starts with the word "C$OMP".

```
C$OMP PARALLEL DO DEFAULT(SHARED),PRIVATE(J)
      DO 100 J = 1, JMAX
      A(J) = B(J)*C(J)
100   CONTINUE
C$OMP END PARALLEL DO
```

Fig. 1. Example of an OpenMP implementation

It is seen that the OpenMP directives are essentially command line options specified within the source code. Parallel version of the code can be compiled using non-parallel Fortran compiler. In non-parallel Fortran compiler, these OpenMP directives are treated as comment lines. Thus the code is portable between parallel and non-parallel compilers.

3.2 OpenMP Implementation on the FPX Code

The FPX rotor CFD code is converted to a parallel version by using OpenMP parallel directives. The code is about 13,000 lines in length, which is a reduced version of the current FPX release version (which is about 30,000 lines in length). The most of the developmental work is performed on NASA-LaRC's Origin 2000 that has a total of 6 processors, while the performance study is made on the U.S. Army's Origin 2000 that has a total of 112 processors.

Parallelization is the process of analyzing sequential codes for parallelism and restructuring them to run efficiently on multiprocessor computers by distributing the computational workload among the processors. Parallelization can be done automatically or manually. Before manual parallelization, Auto-Parallelizing Option (APO) is used to parallelize the code to determine if APO works for the FPX code. APO is a compiler extension that invokes the MIPSpro auto-parallelizing compilers, and automatically generates code that distributes the computational workload among processors. It is found that APO works fairly well on the FPX code when no more than 16 processors are used. The APO automatically parallelizes about 83% of the computational workload. However, when more than 16 processors are used, the APO produces a totally wrong solution. It is also noticed that Auto-Parallelizing Option fails to produce a parallel source code. Therefore, it is impossible to debug the code generated by APO and further hand-code the program for more efficient parallelization, since the source code in parallel version cannot be generated using APO.

As a consequence, manual parallelization through hand-coding become necessary. Among a total of 42 subroutines in the FPX code, the parallelization is done on those subroutines that carry non-negligible amount of computation workload, that is, on the subroutines that carry over 1% of the total CPU time. Manual parallelizing the code using OpenMP is an easy task, sometimes, by simply inserting the parallel directives. Fig. 2 is an example of how a DO-loop can be parallelized using PARALLEL DO

OpenMP Directive, where parallel directive instructs the compiler to parallelize the loop allowing the computational workload to be distributed among multiple CPUs.

```
C$OMP PARALLEL DO DEFAULT(SHARED),PRIVATE(J,K,HU)
      DO 200 J = 1, 1000
      DO 100 K = 1, 250
      HU = A(J) + B(K)
      HT(J,K) = 0.5 * (X(J+1,K,1) - X(J-1,K,1))+HU
100   CONTINUE
200   CONTINUE
C$OMP END PARALLEL DO
```

Fig. 2. Example of how a Do-loop is parallelized

For multiprocessing to work properly, however, the iterations or order of the execution within the loop must not depend on each other. The variable in the loop must standalone and produce the same answer regardless of the order of execution. Loops that dependents on the order cannot be parallelized. If a loop cannot be parallelized in its original form it may be rewritten to run wholly or partially in parallel.

In a Fortran program, memory locations are represented by variable names [5]. To determine whether a particular loop can be parallelized, studying the way variables are used is made throughout many loops in the FPX code. The essential approach to parallelize a loop correctly is to make sure that each iteration of the loop is independent of all other iterations. If a loop meets this condition, then the order in which the iterations are executed in the loop is not important. The iterations can be executed backward or even at the same time, and the answer will still be the same. This property is captured by the notion of data independence [6]. Based on these principles, some parts of the code are rewritten so that the parallelization is done either wholly or partially.

In addition to the PARALLEL DO directive, there are many other OpenMP parallel directives that can be used to parallelize a code. PARALLEL, SECTIONS, and DO directives are also used in this parallel version of FPX code, for example.

4 Parallel Performance Analysis

After successfully converting the FPX code into a parallel version using the OpenMP parallel directives, a series of runs of the code in both serial and parallel versions is made to study the performance of the parallel computation. For parallel version, a different number of CPUs is used for executing the FPX code. The performance analysis is made on varying computational workload by varying the computational mesh size. The mesh sizes of 80x25, 160x49, and 320x97 grid points are used. These cases take from 42 seconds to about 1 hour CPU time in both serial and parallel versions of the code with a single CPU. Both serial and parallel versions of the code

produce the same solutions. The parallel version of the code with one CPU runs as fast as the serial version of the code.

Table 1. Computational performance of the parallel code

No. of CPUs (n)	Problem Size in terms of Number of Grid Points					
	80x25		160x49		320x97	
	CPU Time in Seconds	SpeedUp	CPU Time in Seconds	SpeedUp	CPU Time in Seconds	SpeedUp
1	42	1.0	448	1.0	3,517	1.0
2	25	1.7	247	1.8	1,866	1.9
4	17	2.5	147	3.0	1,178	3.0
8	13	3.2	108	4.1	691	5.1
16	14	3.0	93	4.8	519	6.8
32	14	3.0	93	4.8	496	7.1

Table 1 details the performance results. Up to 32 processors are used for parallel computations. In addition to CPU time, $SpeedUp$ is also given for each run in the table. $SpeedUp(n)$ is defined as the ratio of CPU time with 1 processor to that of n processors, or, $SpeedUp(n) = CPUTime(1)/CPUTime(n)$. If all 100% computational workload were parallelized and there were no communication overhead among processors, $SpeedUp(2) = 2$, theoretically. However, there is always some part of the code's computational workload (such as I/O statements) that has to be carried out serially by a single processor. This sets the lower limit on code CPU run time. The fraction of the computational workload that is parallelized can never be 100%. Moreover, there is less and less benefit from each added CPU after a certain point due to hardware constraints.

The data from Table 1 are presented in Figs. 3-6. Fig. 3 gives CPU time and $SpeedUp$ for the problem with $80x25$ grid points. This case takes 42 CPU seconds with a single CPU. It is seen that up to 8 CPUs can be used efficiently, and after this point adding more CPUs has no benefit at all. Maximum $SpeedUp$ is 3.2 when 8 CPUs are used.

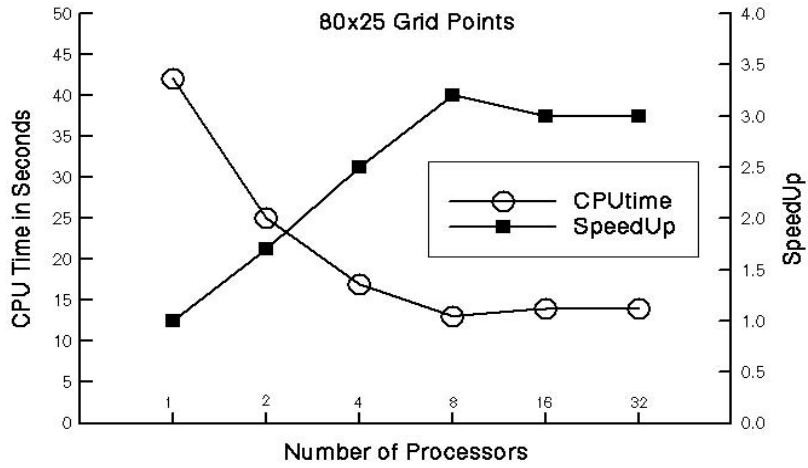


Fig. 3. CPU time and SpeedUp for the problem with 80x25 grid points

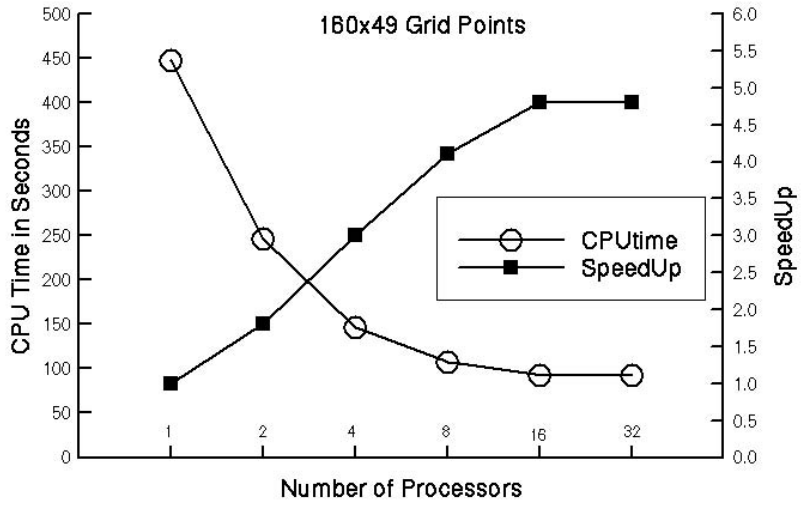


Fig. 4. CPU time and SpeedUp for the problem with 160x49 grid points

Fig. 4 gives CPU time and *SpeedUp* for the problem with 160x49 grid points. This case takes 448 second to execute on one CPU. It is seen that up to 16 CPUs can be efficiently used to achieve a maximum *SpeedUp* of 4.8 due to larger

computational workload than the previous case with 80x25 grid points. Fig. 5 gives CPU time and $SpeedUp$ for the problem with 320x97 grid points. With this mesh size, the code takes about1 hour CPU time to execute on single CPU. It is seen once again from this figure that when the computational workload increases, increasingly more CPUs can be efficiently used. It is seen that all 32 CPUs can be efficiently used for parallel computation to achieve a maximum $SpeedUp$ of 7.1. Using the value of $SpeedUp(2)$ for this case, the fraction of the computational load that is parallelized is calculated to be 95%, which is considered to be substantial.

Finally, Fig. 6 gives a comparison of $SpeedUp$ with varying problem sizes and number of CPUs. The results are self-explanatory. For small computational problem (for example, with 80x25 grid points) less number of CPUs can be efficiently used; with the increase of the computational workload, number of CPUs that can be efficiently used increases also.

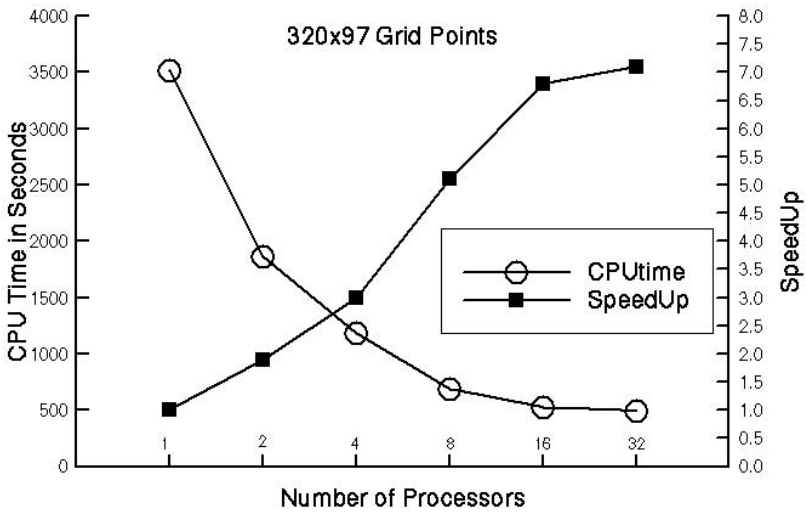


Fig. 5. CPU time and SpeedUp for the problem with 320x97 grid points

5 Conclusion

The eXtended Full-Potential (FPX) helicopter rotor Computational Fluid Dynamics code in its reduced two-dimensional version is successfully converted into a parallel version. The parallel version of the code uses OpenMP directives as parallel programming tool. OpenMP based parallel code is portable and can be compiled with Fortran compiler that supports the OpenMp Fortran standard. OpenMp based parallel code can also be compiled using non-parallel Fortran compiler. As a consequence, no

separate parallel and serial versions of the code are needed, the maintenance cost of the code is thus reduced and the portability of the code between parallel and non-parallel computers increases.

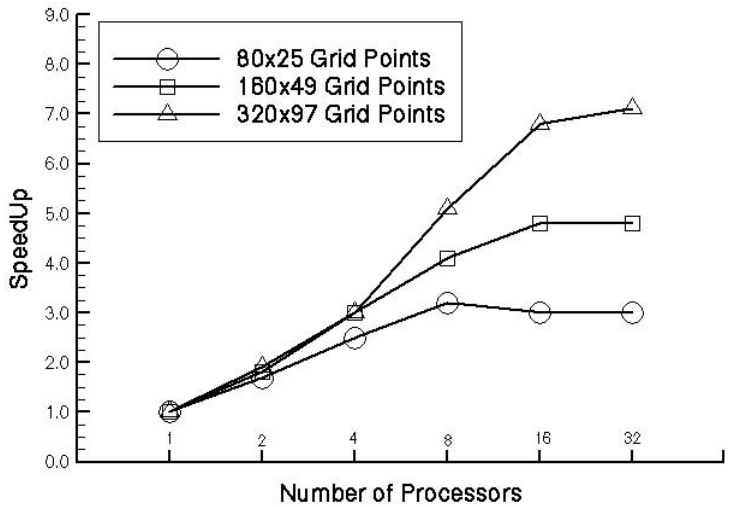


Fig. 6. Comparison of SpeedUp for the problems of all sizes

A performance study of the parallel code is made. From the research presented here, it is concluded that:

(1) Based on the *SpeedUp* results presented here, it is believed that no less than 95% of the computational workload is parallelized; unparallelized part of computational workload may mainly due to I/O statements of the code and the internal grid generator.

(2) For the smallest computational problem tested here, the one with 80x25 grid points, no more than 8 CPUs can be used for efficient parallel computation.

(3) When the computational workload increases, the number of CPUs that can be used efficiently for parallel processing increases also. For example in the case of 320x97 grid points, which is typical for CFD computations, all 32 CPUs can be used efficiently for parallel computation to achieve a maximum *SpeedUp* of 7.1. Scalability increases with the increase of the computational workload.

(4) OpenMP is easy to use and a very efficient parallel programming tool for the present problem. The method is recommended for use in the future work on developing a parallel version of the full three-dimensional FPX code.

Acknowledgement

This work is supported by the NASA Grant NAG-2-1331 from Ames Research Center under FAR program with Dr. Roger Strawn as the Technical Officer and with Dr. Henry Jones of Langley Research Center as a local Point Of Contact. Their support and advice made this work challenging and fulfilling. The computational resources are provided by a grant of HPC time from the DoD HPC Center, ERDC Major Shared Resource Center, and by NASA Langley Research Center.

References

1. Bridgeman, J. O., Prichard, D., Caradonna, F. X.: The Development of A CFD Potential Method for the Analysis of Tilt-Rotors. Presented at the AHS Technical Specialists Meeting on Rotorcraft Acoustics and Fluid Dynamics, Philadelphia, PA, (1996).

2. Strawn, R. C., Caradonna, F. X..: Conservative Full Potential Model for Unsteady Transonic Rotor Flows. AIAA Journal, Vol. 25, No. 2, (1987) 193-198.

3. Breshears, C. P.: Four Different Parallel Architectures- Which One Is Best? The Resource, U.S. Army Engineer Research and Development Center Newsletter, Spring (2000).

4. OpenMP - Frequently Asked Questions. http://www.openmp.org.

5. MIPSpro Fortran 77 Programmer's Guide - OpenMP Multiprocessing Directives. http://techpubs.sgi.com/library.

6. Fortran 77 Programmer's Guide - Fortran Enhancements for Multiprocessors. http:// techpubs.sgi.com/library.

Plasma Modeling of Ignition for Combustion Simulations

O. Yaşar
Department of Computational Science
State University of New York
Brockport, NY 14420

Keywords: Engine Combustion Simulations, Spark Ignition Modeling, Computational Fluid Dynamics, Plasma Properties, Equation of State Data

1 Abstract

Detailed ignition modeling has been traditionally left out of the combustion simulations, mainly due to its relatively small time and length scales that could not be integrated into much larger length and time scales of combustion and fluid dynamics. Although the advancements in computing power has now made possible to integrate ignition and combustion modeling, not many researchers have shown interest due to the wide scope of knowledge it needs in both plasma physics and combustion dynamics. In this paper, we are reporting a preliminary analysis of an integrated ignition+combustion model for both spark ignition (SI) and laser-induced spark ignition (LISI) combustion chambers. A common aspect of both SI and LISI is the use of equation-of-state (EOS) tables at temperatures up to 5 times higher than those assumed in most combustion codes. We report use of an atomic physics code to generate EOS tables needed for our combustion and ignition simulations.

2 Introduction

Fossil-fueled energy supply and end-use combustion systems are the cornerstone of the industrial and commercial sectors of the U.S. economy. In order to optimize the design and operation of combustion systems, a new generation of advanced design tools must be developed to provide the system designer and operator with a fundamental understanding of the interaction of key operating and design parameters. Identified as one of the grand challenge problems, combustion involves many complex phenomena including gas dynamics, chemistry, turbulence, radiative heat transfer, sprays and ignition.

Because combustion occurs at time and length scales where chemistry and turbulence interact, the combustion modeling and simulations face significant scientific challenges. Development of a computer system that can fully resolve, numerically, all aspects of the operation of a combustor may not occur in the foreseeable future. Instead, the approach proposed by the combustion community [1] involves a selective and systematic development of submodels that allow

V.N. Alexandrov et al. (Eds.): ICCS 2001, LNCS 2073, pp. 1147–1155, 2001.
© Springer-Verlag Berlin Heidelberg 2001

"bootstrapping" from the microscopic regimes to the device-modeling regime. In order for this approach to be successful, new knowledge in the areas of chemistry, turbulence, materials science, mathematics, and computer science need to be discovered.

Here, we focus on development of an ignition submodel. If a reliable, and reproducible ignition is found, this will help scientists sort out effects of early ignition on combustion. This might even help studies of turbulence and turbulence-combustion interactions which are current concerns of researchers. We will limit our focus to early flame formation and its plasma characteristics.

The spatial and temporal resolutions needed to resolve the ignition source are beyond the capability of current combustion models. Ignition takes place in a very tiny space (microns) and in a quick time-frame (nanoseconds), however it has a lasting effect on what follows. Early flame formation and propagation has a vital impact on the system performance and emissions.

3 Plasma Properties of Ignition

The thermodynamic properties of hot plasma are often required in hydrodynamic and radiative energy transport of astrophysical and fusion plasmas [2]. Although the physical conditions in such cases seem to be far from those of an internal combustion engine, researchers have reported very high teperatures (up to 60,000 Kelvin) and densities during the ignition process [3]. During the spark breakdown phase, the ion density increases rapidly to values of some $10^{19} e/cm^3$ which leads to a significant energy loss for accelerated electrons due to Coulomb collisions with the ions [11]. The frequency of such collisions determines the electricial resistivity (η), which is dependent on the degree of ionization of the background gas. We know from classical transport theory that [7,10,11] conductivity (or resistivity) includes both the effects of Coulomb collisions and electron-atom collisions. The expression used is [11]

$$\eta = \eta_W + \eta_F \tag{1}$$

where η_W is (Weak) Coulomb collision, and η_F is (Full) electron-atom collision term as

$$\eta_F \equiv (5.799 \times 10^{-15}) \cdot Z(T) \cdot ln\Lambda \cdot T^{-3/2}$$

and

$$\eta_W \equiv \frac{m_e^{1/2}}{e^2} \frac{\sigma_{ea}(T)}{\alpha(T)} T^{1/2}$$

Here, σ_{ea} is the electron-atom collision cross section, m_e is the electron mass, e is the electron charge, Z is the average ionic charge state, T is the plasma temperature, and α is the degree of ionization of the plasma. The degree of ionization is a strong dependent of the plasma temperature. When the degree of ionization falls below 0.001, the behavior of η is controlled by η_W. For α greater than 0.1, the dominant term is η_F.

Calculation of the average charge (Z), the electron-atom collision cross section (σ_{ea}), and the degree of ionization (α) is neccessary for computing plasma resistivity which determines how much heat is deposited in the early ignition phase. The average charge (Z) is also needed for computing the equation-of-state (EOS)

$$P = (1 + Z)nkT,$$

which is needed to bring closure to the governing equations [4]. In most, cases, this EOS relation is sought between the specific internal energy (e), the number density (n), and the temperature (T).

We use IONMIX [12] to compute the steady-state ionization and excitation populations for a mixture of up to 10 different atomic species. The thermodynamic properties of plasma, such as the specific energy, average charge state, pressure, and heat capacity are also calculated. One can obtain the radiative absorption, emission, and scattering coefficients, however we are not using these to solve any radiation transport equations at this stage of our work.

The ionization populations for IONMIX are computed for steady-state conditions. The atomic processes considered are collisional ionization, radiative recombination, dielectronic recombination, and collisional recombination. IONMIX uses ionization potentials given at Nuclear Data Tables [12] for about 15 elements. The user must supply an ionization potential for elements outside of this range.

Several aspects of ignition modeling are being considered here are: the amount of energy deposition, the resistivity model, and the EOS data at high temperatures. Most of these issues are common for both SI and LISI, however the EOS data goes to the heart of our approach. The energy deposition for SI is computed through the solution of electromagnetic equations and we have presented this, at least for 1-D at an earlier publication [4]. The energy deposition for LISI is rather an estimate base on experiments [15]. All of our SI and LISI computations have been integrated as sub-models into the KIVA-3 engine combustion code [13]. To our knowledge, these SI and LISI models were first attempts to examine the influence of a computed ignition model, instead of a semi-emprical approach, on combustion output. Here, we are attempting to measure the influence of a well-expanded EOS data for both SI and LISI. Our database for chemical reactions (kinetic and equilibrium) is currently following the data available in KIVA-3. However, this database need to be expanded to higher temperatures than currently available.

4 Results

Our one-dimensional spark-ignition model described in [4] along with expanded EOS tables described here and in [12] has been integrated into the standard KIVA-3 code [6,13]. We examined the combustion output (i.e., temperature, NO_x density) against ignition energy variations. The new EOS data obtained for high temperatures via IONMIX has been used to check the variations in

combustion output. This section lists preliminary results for both SI and LISI cases.

4.1 SI

Although far from representing all spark phases in [14], the spark discharge profile taken from experimental data by [5] suffices for our initial tests involving the engine configuration in 1.

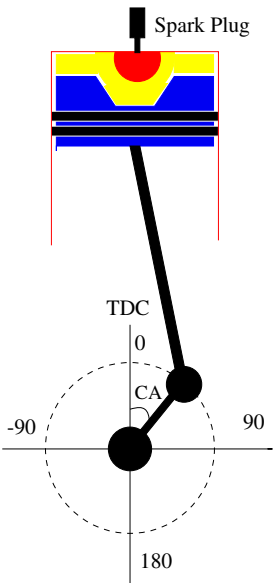


Fig. 1. Schematic of an engine.

We examine the influence of added plasma properties and equations in our simulations. These new properties include 1) computation of ignition energy source via Maxwell's equations [4], 2) use of new EOS data from IONMIX, 3) a new resisitivity model to compute spark current density. A comparison is made to a test case using the standard KIVA-3 without any such plasma properties.

These results have been obtained on a baseline engine with a spark plug centrally located at the cylinder head. Temporal and spatial data was obtained at every 5 degree crank angle for a computational mesh of approximately 5,500 elements. The engine specifications can be found in [6,13]. The enormity of the data makes visualization a necessity for scientific interpretation. The standard KIVA-3 output was converted to a format accepted by the AVS post-processing software.

We include a summary of our observations via Fig. 2 and 3 for different fuel amount and varying spark current. In these figures, we present peak values of

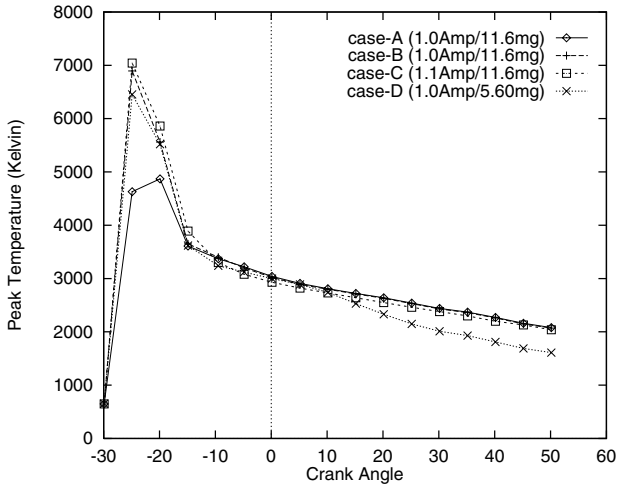


Fig. 2. Temperature at the ignition center

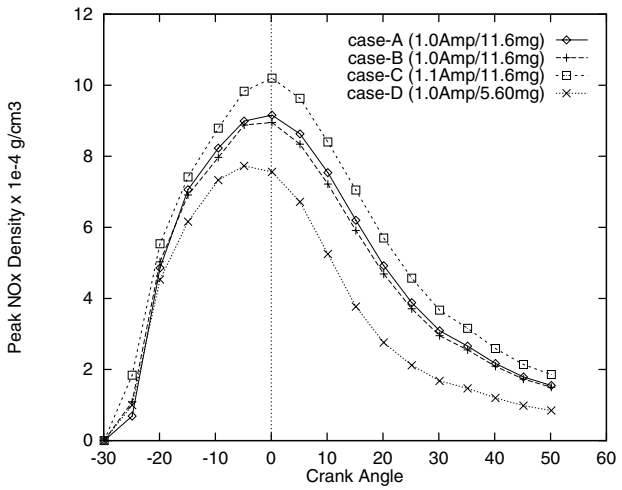


Fig. 3. NO_x density at the ignition center

temperature (T), and NO_x density at the ignition region. It is obvious that our plasma-enhanced simulations (Case B,C,D) show a more elevated peak temperature than the classic KIVA-3 run (Case A) with a simpler ignition source and limited EOS data. The difference between Case A and B (which have the same fuel amount and the same spark current) is almost seventy percent. This difference is significant enough to lead to an entirely different NOx and fuel density profiles. We expect that a significant part of the temperature difference comes from the use of new EOS tables that enable us to move into a high temperature region rather than cutting it off at the 5,000 Kelvins as the KIVA-3 code does.

Contrary to several assumptions in previous modeling efforts, temporal and spatial variations of the spark plasma properties are important source of influence on the flame dynamics and subsequent combustion. These results justify a more comprehensive approach to account for the high temperature aspects of ignition. They also indicate that the evolution of ignition and its interaction with combustion should be taken into account.

4.2 LISI

Although experimental data from LISI clearly shows spatial and temporal variations [15], we assume a fixed energy source during the ignition window. Furthermore, the source term is $\Delta e = (P/\rho) \cdot \Delta t$, where P is the laser energy deposition rate ($ergs/s.cm^3$), ρ is the total density, and Δt is the duration of the laser pulse. Although the laser lasts only 5-10 nanoseconds, Δt here might be as low as as one nanosecond. In that case, the sum of all the Δe values during the ignition window would give the total ignition source energy deposition. The use of small Δt is to prevent numerical instabilities that could arise from a high deposition rate. The laser energy deposition rate used here is $5 \times 10^{16} ergs/s.cm^3$, which deposits about a fraction of a $Joule$ energy in the combustion chamber in the 5-10 nanosecond timeframe.

We examine the effect of an expanded EOS data on the combustion output. As it is in the SI case, the temperature in the combustion chamber is the most affected physical quantity. A comparison of temperature, NO_x Density, and the Fuel Density for two cases is presented in Figures 4,5, and 6. One of these two cases represents original EOS data, which is below 5,000 Kelvins. The other case is for expanded EOS data beyond 5,000 Kelvins. It seems the temperature difference between these two cases leads to a different level of fuel consumption, and NO_x production. In the expanded EOS case, the temperatures in the ignition region are allowed to rise, therefore leading to more fuel consumption and NO_x production. In the figures shown here, the NO_x density is much higher for limited EOS case in the central region due to a higher fuel density at that time compared to expanded EOS case. In the expanded EOS case, more fuel is burned earlier. The produced NO_x leaves the center of chamber. The overall NO_x production is therefore still higher for the expanded EOS case.

Since the effect of temperature is so high on fuel consumption and NO_x production, among other things, it is important that we conduct an accurate solution, particularly in the use of EOS data. If the chamber temperature is

much higher than what we believed in, then this will lead to an underestimation of pollutants and wear and tear of chamber equipment.

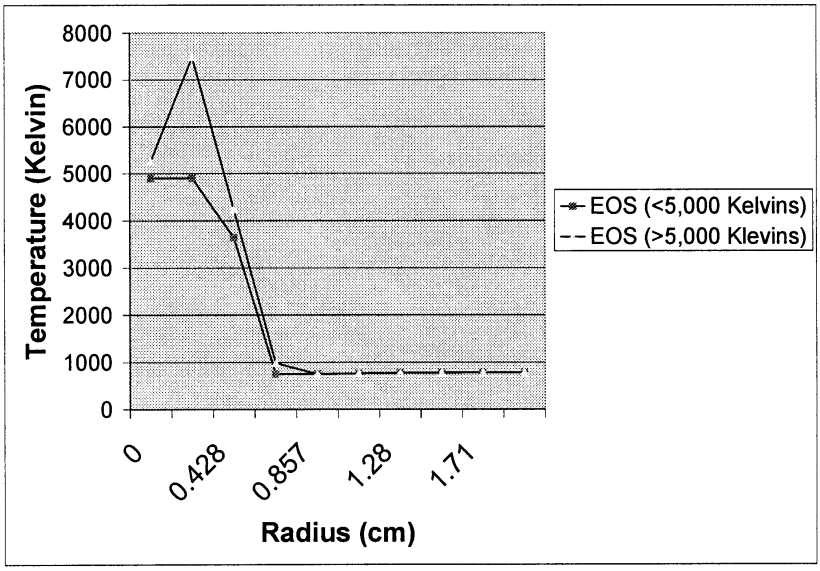


Fig. 4. Temperature at Crank Angle=-4.99

5 Conclusion

It is noted that temperatures in the ignition area are many times higher than those in the rest of the combustion chamber. Detailed ignition modeling is needed to enhance accuracy of combustion simulations. The equation of state (EOS) data as well as the reaction rate tables need to be updated to include temepratures up to at least $10,000 - 20,000$ Kelvins. The use of an tomic physics code (IONMIX) here has been justified as there is a significant difference in results when we use an expanded EOS data. We will report further results in future publications. The ignition sub-model and the expanded EOS tables will be modularized to be used with other combustion codes.

References

1. U.S. Department of Energy, Strategic Simulation Initiative, http://www.er.doe.gov/ssi.
2. D. Mihalas and B. Mihalas, Foundations of Radiation Hydrodynamics, Oxford University Press, 1984.
3. R. Maly, "Spark Ignition: Its Physics and Effect on the Internal Combustion Engine," in Fuel Economy, Editor: J. H. Hilliard and G. S. Springer, Plenum Press, 1984.

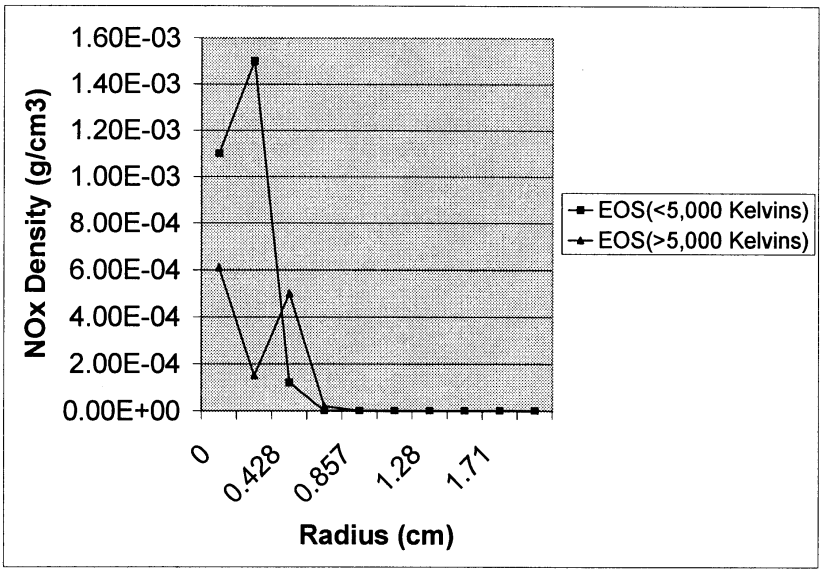


Fig. 5. NOx Density for LISI at Crank Angle=-4.99.

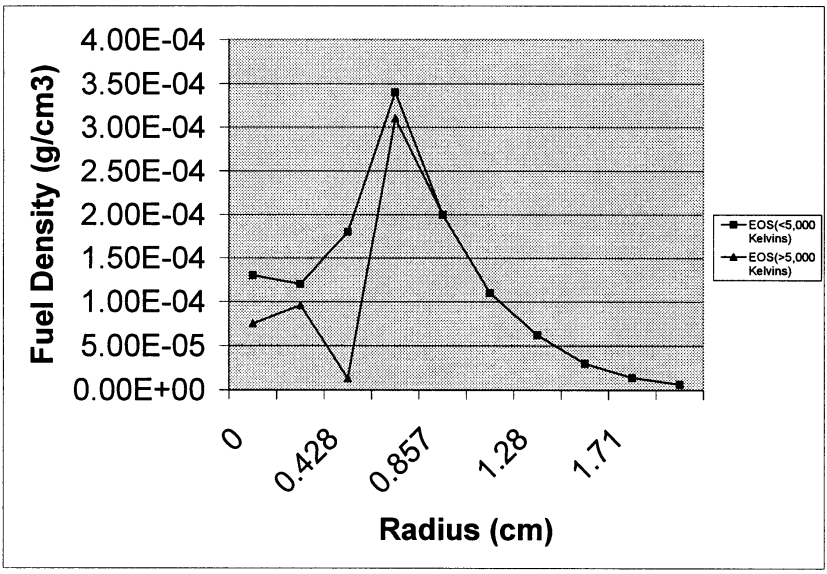


Fig. 6. Fuel Density for LISI at Crank Angle=-4.99.

4. O. Yasar, "A New Spark Ignition Model for Engine Simulations," *Parallel Computing*, **27**, 1 (2001).
5. Sauers, D. D. Paul, J. W. Halliwell, C. W. Sohns, Energy Delivery Test Measurement System Using an Automotive Ignition Coil for Spark Plug Wire Evaluation, Patent Disclosure ESID No. 1774XC, S-83,831.
6. A. A. Amsden, "KIVA-II: A Computer Program for Chemically Reactive Flows with Sprays," Los Alamos Technical Report LA-11560-MS, 1989.
7. O. Yasar and G. A. Moses, "Explicit Adaptive Grid Radiation Magnetohydrodynamics," *J. Comput. Phys.*, **100**, 1 (1992) 38.
8. W. F. Hughes, F. J. Young, The Electromagnetodynamics of Fluids, John Wiley, N. Y. 1966.
9. J. D. Jackson, Classical Electrodynamics, John Wiley & Sons, New York (1975).
10. J. J. Watrous, G. A. Moses, and R. R. Peterson, "Z-PINCH - A Multifrequency Radiative Transfer Magnetohydrodynamics Computer Code," Technical Report UWFDM 584, University of Wisconsin-Madison, Fusion Technology Institute, 1985.
11. S. V. Dresvin (editor), Physics and Technology of Low Temperature Plasmas, Iowa State University Press, Ames, IA (1977).
12. J. J. MacFarlane, P. Wang, and O. Yasar, "Non-LTE Radiation from Micro- fireballs in ICF Target Explosions," Bull. Amer. Phys. Soc., 34, 2151 (1989).
13. A. A. Amsden, "KIVA-3: A KIVA Program with Block-Structured Mesh for Complex Geometries," Los Alamos Technical Report LA-12503-MS, 1993.
14. R. Stone, Introduction to Internal Combustion Engines, SAE Publications, pp 150-153, 1993.
15. P. X. Tran, "Laser spark ignition: Experimental determination of laser-induced breakdown thresholds of combustion gases," Optics Communications, 175, (2000) 419-423.

Computational Science Education: Standards, Learning Outcomes, and Assessment

Osman Yasar
Department of Computational Science
State University of New York
Brockport, NY 14420

Abstract

We have entered a new phase in the growth process of computational science. The first phase (1990-2000), which coincides with the federal high performance computing and communication program, can be named as the *recognition phase,* at the end of which there was a general agreement to accept computation and computational science as a distinct methodology and discipline. The recognition started at the doctorate level and moved down to at least the baccalaureate level and even to a few high schools. This first decade of growth period saw at least one stand-alone computational department, one stand-alone school at the dean level, and a program to train high school teachers. The second phase (2001-2010), which coincides with the federal information technology program, will witness curriculum standardization at all levels, perhaps accompanied with an accreditation mechanism for future programs. It is important to assess student success in the new programs. In this paper, we will address learning outcomes and assessment techniques, followed by a brief account of research-curriculum integration at our institution. We will also give a brief overview of computational science and engineering.

1. Overview

Professional societies such as SIAM (www.siam.org), IEEE Computer Society (www.ieee.org/computer), ACM (www.acm.org), AMS (www.ams.org) and Society of Computer Simulation (www.scs.org) have all undertaken major initiatives to organize annual conferences on computational science and engineering (CSE). There are also new professional societies putting computational science at the center of their activities. These include Society of Computational Biology and Society of Computational Economics, among others. The number of web sites for CSE programs, research centers, government labs, and industrial settings has grown by an order of a magnitude. A search for "computational science" over the Internet gets several hundreds to thousands of hits and links. A recent resulted in the following number of hits: GoTo (240), LookSmart (130), Lycos (46,045), HotBed (466,500), and AltaVista (65,447). At least the first 500 of such links were reviewed and found to be very relevant to the search topic.

Computational science and engineering has emerged as a new discipline in the past decade. At the core of this development is a dramatic increase in the power and use of computers. Capitalizing on advances in computing technology, new methods and programming tools were developed to solve problems that were not in our reach before. As much of an interdisciplinary program as computational science is, it is also a discipline of its own due to: 1) the amount of knowledge involved in its presentation

V.N. Alexandrov et al. (Eds.): ICCS 2001, LNCS 2073, pp. 1159–1169, 2001.
© Springer-Verlag Berlin Heidelberg 2001

to colleagues and students, 2) the amount of scientific literature (journals and conferences) devoted to this topic, and 3) the work and service involved for further recognition, establishment, and success of computational culture in educating new generations.

Computational science has two intermingled contexts attached to it: 1) *science of computing,* and 2) *science that is done computationally.* The CSE field investigates computational techniques that are common to many applications; therefore it focuses on the art/science/engineering of computing. The applications that use computing are many, ranging from basic sciences to engineering and industrial problems. All these compute-bound scientific and industrial problems form a bond together with many exchanges and commonalities among each other. Under the umbrella of *computational science,* one can find *computational biology, computational physics, computational chemistry, computational finance, and computational mechanics,* and so on. However, the discipline of computational science does not necessarily cover core knowledge and experience of all these sciences; it only covers their computational aspects. Therefore, computational science serves not a replacement to any of the science disciplines, but as a bridge between sciences, engineering, computing, and mathematics [1].

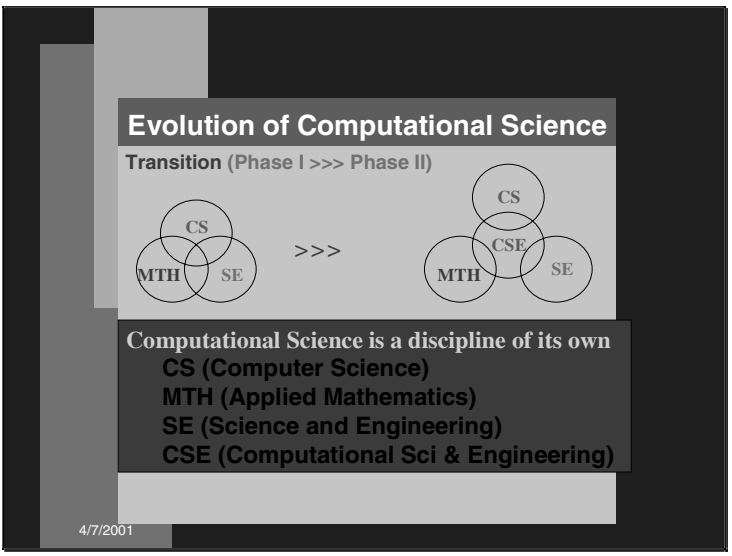


Fig. 1. Evolution of Computational Science and Engineering

There is a natural overlap of CSE with other sciences, however CSE has a core knowledge base of its own. Computational science and computer science have common concerns when it comes to performance of computer hardware/software and anything related to optimizing one's application on computers. Computational science and mathematics have common concerns when it comes to applied math techniques to numerically solve partial differential equations. Finally, computational science shares

concerns of application areas (such as physics, engineering, chemistry, biology, earth sciences, business, art) in terms of finding a computational solution to complement both theoretical and experimental efforts. In some cases, what can be accomplished by computation cannot be done otherwise. Physical systems that are too small, too big, too expensive, too scarce, and not accessible (experimentally) are being modeled on computers with a great deal of success. Examples of probing atomic systems (too small), studying earth and universe (too big), weighing impact of an asteroid on earth and studying internals of an engine piston (not accessible) are just a few. Computer visualization of such systems has also created a new way of gaining insight that otherwise would not have been discovered.

A field that involves so much information and draws upon knowledge in other areas also needs time and attention to identify and study techniques common to many applications. It also needs full-devotion to the study of performance of computer hardware/software as well as of computational methods and tools that otherwise might not have been studied. When one considers all these non-overlapping and overlapping components, the field basically becomes a discipline of its own. At the heart of the field is the study of common computational techniques, and unless there is a full devotion to this study, the field cannot advance very quickly. The transition of CSE from a mere overlap of computer science, math, and applications to a field with its own identity and knowledge base is now taking place, as illustrated in Fig. 1. Although we did not encounter a consensus on this latest development earlier, we now note similar views published recently by our colleagues [3]. Scientists and students in this area have a unique identity as computational scientists and engineers who have gathered a combination of practical knowledge in computing, mathematics and applications.

2. Student Learning Outcomes

At SUNY Brockport, we offer both undergraduate (B.S.) and graduate (M.S.) degrees in computational science. Our program started in the Fall of 1998 and was transformed into a department after two successful years of operation. We have about 50 students enrolled in the program and our first graduates have actually hit the job market. To our knowledge, we were the first undergraduate program in computational science, and now perhaps the only department in this field. Having no precedence before us, we have struggled with many issues including curriculum development, faculty career development, tenure guidelines, recruitment, placement, and documentation. Our curriculum was revised recently and we expect minor revisions in the future. We are still developing new courses, particularly in the area of computational applications.

In an effort to encourage standardization, we published our assessment of the elements of a typical computational science education [1,2]. As mandated by our college policies at SUNY Brockport, we now have identified, in a more concise way, the students learning outcomes (SLO) as listed below. The task before us will enable us to document in detail all measurable aspects of a computational science education, including the seven SLOs we have identified below:

(a) Learning the use of computers and computational tools,
(b) Learning high-level languages and the use of high performance computers,
(c) Obtain a knowledge of applied math and computational science methods,
(d) Learn basics of simulation and modeling,
(e) Learn how to visually interpret and analyze data after a simulation is completed,
(f) Learn about at least one application area to apply acquired computing skills to,
(g) Learn to communicate solution methods and results.

The challenge, of course, is to find ways to measure whether these outcomes have been achieved. In Table 1, we list the first SLO and relevant course objectives used to measure its outcome. Course titles and descriptions in our program can be found at http://www.cps.brockport.edu. The courses referenced in this table, however, are listed here again as:
CPS 101 Introduction to Computational Science
CSC 120 Introduction to Computer Science
CPS 201 Computational Science Tools I
CPS 202 Computational Science Tools II
CSC 203 Fundamentals of Computer Science
CPS 303 High Performance Computing
CPS 433 Scientific Visualization
CPS 602 Advanced Software Tools

Table 1. SLO # 1: Learning the use of computers and computational tools

Course	Course objective
CPS101	(1) To learn about functions, their uses and representations. (2) To learn about behaviors of functions and the rate of change (3) To find a functional relation based on behavioral relation using numerical integration (4) To learn the fundamentals of FORTRAN 77 (5) To learn the UNIX operating system and become comfortable in a UNIX working environment. (6) To solve simple real world problems, using numerical solutions, programming, and: (a) differentiation (rate of change problems), (b) integration (area and volume problems), (c) linear regression. (7) To identify a few industrial and scientific problems and their computational solutions
CSC120	1) To learn the internal workings of computers: (a) hardware components such as CPU, memory, disk storage, peripheral devices, etc., and measures of performance, (b) gates and simple circuits such as flip-flop, (c) data types and their internal representations, (d) operating systems and their components, (e) the need for hardware and software standards, and (f) networks and the Internet. 2) To learn high-level and low-level language concepts 3) To learn program execution in terms of machine instructions 4) To learn elementary concepts and syntax of C/C++ programming languages 5) To learn algorithms: (a) examples of some simple algorithms, (b) designing and testing algorithms, (c) translation from the problem domain to the programming domain.

CPS201	(1)	To review programming languages: C++ & F77
	(2)	To learn about abstract data structures in C++ & F77
	(3)	To learn how to use a symbolic manipulation tool (MATHEMATICA) to model simple problems.
	(4)	Use the graphic capabilities of MATHEMATICA to assist in the analysis of simple models
	(5)	To learn the use of LaTeX and the concise, clear presentation of results from simulations.
	(6)	To learn how to use the UNIX operating system (directories, file editing, program compilation).
CPS202	(1)	To learn basic principles of programming in Fortran 90.
	(2)	To learn the use of the MATHLAB software tool including a) language constructs, b) 2d graphics routines (x-y plots, scatter plots, contour plots) c) 3d graphics (surface and mesh plots, line graphs).
	(3)	To learn the use of the Advanced Visualization System (AVS) software tool. This includes a) generating simple plotting interfaces, b) generating a GUI that can interface with external C and Fortran routines, c) working with 2 and 3d graphics as outlined above.
	(4)	To learn about mathematical algorithms: a) random numbers, b) Gaussian elimination, c) Fast Fourier
	(5)	To learn how to use industry standard computational libraries (LAPACK, ATLAS).
CSC203	(1)	To learn fundamental computer science concepts and programming in C++
	(2)	To learn about sorting and searching techniques
	(3)	To learn about files, trees, recursion, graphs, pointers, and classes
CPS303	(1)	Become proficient in the basic use of the MPI message passing library for solving simple problems in parallel.
	(2)	Become familiar with programming in a batch processing environment (compiling, submitting jobs, checking job status, queues).
	(3)	To learn how to decompose a problem in a manner suitable for efficient parallel implementation including communication structuring.
	(4)	To learn how to evaluate the performance of a parallel algorithm (in terms of speedup, efficiency and scalability) and how to modify a given algorithm for improved performance
CPS433 CPS533	(1)	To learn the value of graphic visualization in the context of large or highly complex data sets.
	(2)	To learn the use of the graphical capabilities of various graphical software packages (MATHLAB, MATHEMATICA, XMGR).
	(3)	Learn to develop graphic applications specific to a given discipline using the Advanced Visualization System (AVS) tool.
	(4)	Learn how to interpret simulation results and detect possible errors in data for model simulations arising from the physical sciences.
	(5)	Develop GUI tools using AVS and MATHLAB.
CPS602	(4)	To learn the use of standard parallel programming libraries (ScaLAPACK, PETSc,MPI).
	(5)	To learn the use of various problem solving environments (Netsolve) and parallel tools.
	(6)	To learn how to use grid generation. Algorithms (including automatic mesh generation) to solve models of partial differential equations.

The course objectives listed above need to be measured in each category and subcategory. The method of measurement is homework assignments and tests. The standard for success has been set to 80 % of the assignments and tests in our program. Further details of assessment techniques is available via the author.

For more information on the remaining SLOs, the reader can contact our department directly at cps@brockport.edu, or visit http://www.cps.brockport.edu.

The field of computational science is very dynamic. It is new and still being defined. Since it is a technology (computer hardware and software) oriented field, the content and the curriculum is often being updated. Our program started only 2 years ago, yet we have already done a major revision to our curriculum. Further, but less radical, revisions are expected in the next 3-5 years. In the next 2 years we will still be in a course development mode at both the graduate and undergraduate levels. The content of these courses and the success of our curriculum will greatly depend on our faculty members' updated knowledge about the computer hardware/software, latest mathematical methods and the computational tools in the market place and the literature. Since the computer technology changes radically every 12-18 months, we must quickly adapt to new technology so that our graduates can adjust more easily to the job market. In one respect, we are more technology dependent than the field of computer science where the basics of computing are taught. In computational science, the computing must be put in the context of applications that are driving the market. This dynamic aspect of computational science requires faculty members to be well connected to the research and the industrial community so they can teach updated material and provide timely advice.

3. Research-Curriculum Integration

The research interests in our department cover engineering and scientific aspects of different areas such as engine combustion, fluid dynamics, molecular dynamics, and weather modeling, yet they all use a set of common tools, namely computation, simulation and visualization. The expertise of faculty members in different industrial and scientific areas is brought into the classroom to introduce students to these topics in a hands-on way where they can learn more effectively by simulating systems of their choice. The collective experience by our team on common tools such as computing, numerical methods, parallel programming, and visualization is also brought to the classroom to advance knowledge of students in engineering and computing sciences.

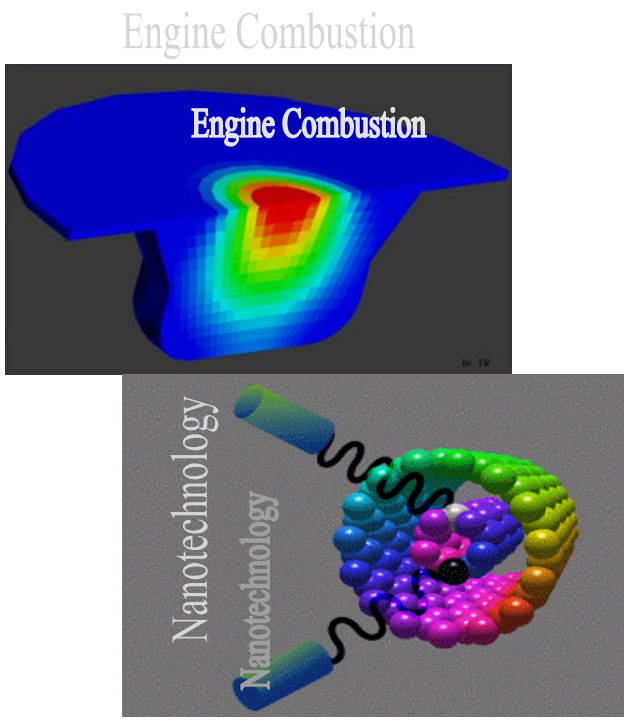


Fig. 2. Two examples of computational science applications at SUNY Brockport

3.1. Engine Combustion

Combustion has been identified as a major study area under the 1999 Presidential Initiative IT-2 (Information Technology Initiative II). Strict regulations on air quality require cleaner engines. Development of full-scale and full-physics (flow, combustion, plasma, spray, radiation) combustion codes is critical for the success of these new programs. Although high-fidelity simulations of internal combustion engines and industrial burners require computers 10,000 times faster than current personal computers, the ability to simulate reasonably representative systems has gone well beyond the circle of a few national labs. For example, a publicly available engine code, KIVA [4], from Los Alamos National Lab can now be run on personal computers. The graphical software to display results in a visual way has also enhanced our ability to understand results and shortened the time to model and analyze engineering systems.

Our combustion team has demonstrated research expertise in engine combustion simulations as well as computational aspects of computer science and mathematics [5-9]. Our version of the engine code KIVA has been referenced by many as the only scalable version that can do multiple engine cycle simulations due to its capacity to simulate multi-million level mesh computations in a reasonable amount of time. This code has been used for joint collaborations with industry. Another aspect of our engine work is the marriage of plasma hydrodynamics [5-6] with engine combustion. We have taken a dramatic approach to fully simulate the interaction between

combustion dynamics and spark ignition. This work resulted in two major CRADAs with industry. Previous approaches to spark ignition dynamics and its effect on combustion dynamics had been limited to crude approximations, yet the amount of spark energy into the combustion chamber is the most critical element of engine operation (for spark-ignited engines). An accurate computation of spark energy deposition into the combustion chamber requires a time-dependent feedback between sparking event and gas dynamics, though this requires solution of both electromagnetic and fluid flow governing equations at a much finer time-scale (nanoseconds) than typical flow simulations.

Integration of scalable engine combustion simulations with advanced visualization techniques has brought our combustion research to a level to be integrated into both engineering and computer science classrooms. Use of AVS (commercial product for visualization) and EIGEN/VR (from Sandia National Labs and Oak Ridge National Lab) to visualize engine simulations have proven a valuable tool for engine designers and future computational scientists and engineers.

The experience gained by our team in computational engine simulations is used in several courses, including Simulation and Modeling, Supercomputing Applications, Deterministic Dynamical Systems, and Scientific Visualization. The engine combustion code to be experimented with in these classes is called KIVA [4]. It has been modified and enhanced by many research groups and these modifications have been presented and examined at the KIVA International Users Group meetings during the Society of Automotive Engineers Convention. Versions of KIVA have been around for years since 1985, but its full potential to teach students about engines has never been utilized due to limited computer resources. Yet, such tools are very common in industry and students both at engineering departments and at computational science programs should be given the opportunity to learn industrial engine design through combustion simulations and engine visualization techniques.

The parallelization of KIVA [7] is also subject of a course within our computational science and engineering curriculum as it teaches about domain decomposition, computation/communication overlap, and effective programming. The availability of the scalable KIVA-3 presents great potential for a computational scientist and engineer to learn about industrial requirements of engines in a class environment by doing different engine simulations for sensitivity analysis. Post-processing is also directly applicable to a course in scientific visualization.

Computational Chemistry

Molecular simulation is an active area of research and scientific application that requires computational techniques used in many areas of study. Since chemical reactions (dynamical processes such as phase separation, crack propagation in brittle materials, and so on; fluid flow; and other phenomena) cannot be directly observed at the experimental level, molecular simulations of these processes and visualization of the results can provide a wealth of valuable information. In addition to dynamical processes, many materials properties can be calculated from molecular simulation data. All of these methods have found uses in basic research in chemistry, biology,

and other fields and in important applications such as rational drug design. In order to increase the range of applicability of molecular modeling techniques, advances in both computational power and algorithms are required and remain an active area of research. Classical and quantum mechanical molecular simulation techniques provide ample opportunity for illustrating computational methods such as 1) numerical solution of deterministic partial differential equations, predictor-correctors, symplectic integrators, 2) ensemble methods, thermodynamic averages, fluctuations, correlation functions, transport coefficients, 3) constraint dynamics, 4) optimization, conjugate gradient and other commonly used methods for molecular mechanics, 5) random number generation and Monte Carlo techniques: random walks, importance sampling, solution of differential equations with diffusion terms, 6) specialized load balancing and domain decomposition strategies.

Our computational chemistry group has extensive background in chemical physics, numerical methods, and molecular simulation methods. In the past five years, they have developed several generalizable, robust, specially portable algorithms for molecular dynamics, molecular mechanics, and quantum Monte Carlo simulations [10-15]. Applications of these new methods include polymer science; nanotechnology; molecular fluid flow; and classical-quantum correspondence in many-body systems. The computational chemistry research interests in our program also include few-body quantum mechanical calculations and the quantum theory of angular momentum.

Molecular simulation is eminently suited for classroom presentation, both at the graduate and undergraduate levels, and it forms an important part of the computational science curriculum. In Simulation and Modeling course students gain the basic knowledge for writing complete simulation programs and for analyzing the results. In Deterministic Dynamical Systems, in Stochastic Dynamical Systems, and in Supercomputing and Applications courses, students learn in more detail topics such as optimization, numerical solution of partial differential equations, Monte Carlo methods, random number generation, use of physical principles for code validation, parallel programming strategies, benchmarking, and specialized load balancing and domain decomposition techniques. These skills are commonly used in climate research, automobile design, environmental research, and a host of other important scientific and engineering applications.

Weather Modeling

Today, the science of weather forecasting relies heavily on numerical weather predictions. Every day, supercomputers at the National Centers for Environmental Prediction (NCEP) run at least five different NWP models; each model with its own unique set of equations representing atmospheric dynamics. In tandem with the increase in computational power, the level of detail and the sophistication of the parameterization of physical processes in these NWP models have increased. However, the translation of these advances into improved weather forecasts has been slow. Weather forecasters, trained professionals who interpret NWP model output, generate public forecasts after comparing an increasingly complex set of models and reconciling discrepancies between them. Without the benefit of a fully integrated

visualization tool, these forecasters cannot realize the full potential of the NWP system.

Virtually all software tools used by weather forecasters to display NWP model outputs generate two- dimensional maps, with some capacity to overlay more than one field. These tools do not fully utilize the wealth of information provided by NWP models; there is tremendous room for improvement. In fact, few, if any, operational forecast offices use three- or four-dimensional visualization tools. The general public find forecast maps difficult to comprehend or of limited use. In both forecasting and public presentation, present day electronic media provide limitless opportunities.

To learn weather forecasting, students majoring in Meteorology at SUNY Brockport use outputs from the NWP models generated at NCEP. A critical step in the forecast process is the ability of forecasters to interpret NWP model outputs in an accurate and timely manner. Roebber and Bosart [16-17] have demonstrated that experience is an essential element in the forecast process, and that human judgement allows skilled forecasters to issue forecasts that are superior to raw NWP model outputs. After proper training, students in two of the Earth Science courses (ESC 312 Weather Forecasting and ESC 490 Weather Briefing) are placed on a rotation to forecast different scenarios with and without the benefit of the four-dimensional data visualization. The overall and individual student forecasting performance is tracked, and the degree to which forecast skills benefited from this new technology is assessed. Every semester, anywhere from 150 to 200 students, mostly college freshmen, register for three of the introductory level courses in the Earth Science department (ESC 102 Elements of Geography, ESC 210 Weather I, and ESC 211 Weather II).

References

1. O. Yasar, *et al.,* “A New Perspective on Computational Science Education,” *IEEE Computing in Science and Engineering,* Vol. 2, No. 5, 2000.

2. O. Yasar, “Computational Science Program at SUNY Brockport,” *Proceedings of First SIAM Conference on Computational Science and Engineering,* September 21-24 2000, Washington, D.C.

3. Graduate Education for Computational Science and Engineering, SIAM Working Group on CSE Education, http://www.siam.org/cse/report.htm.

4. A. A. Amsden, "KIVA-II: A Computer Program for Chemically Reactive Flows with Sprays," Technical Report, LA-11560-MS, Los Alamos National Laboratory (1989).

5. O. Yasar, “A New Spark Ignition Model for Engine Combustion Simulations,” *Parallel Computing,* Vol. 27, No. 1-2, 2001.

6. O. Yasar, "A Scalable Algorithm for Chemically Reactive Flows," *Computers and Mathematics*. Vol. 35, No. 7, 1998

7. O. Yasar and C. Rutland, "Parallelization of KIVA-II on the iPSC/860 Supercomputer," in *Parallel Computational Fluid Dynamics*, Editor: R. B. Pelz, A. Ecer, and J. Hauser, North Holland (1993), p. 419-425.

8. O. Yasar and G. A. Moses, "Explicit Adaptive Grid Radiation Magnetohydrodynamics," *J. Computational Physics*, 100, 38 (1992).

9. Y. Deng, R. A. McCoy, R. B. Marr, R. F. Peierls, and O. Yasar, "Molecular Dynamics on Distributed-Memory MIMD Computers with Load Balancing," *Applied Math Letters*, 8 (3), 37-41 (1995)

10. Robert E. Tuzun, Donald W. Noid, and Bobby G. Sumpter, "Automatic differentiation as a tool for molecular dynamics simulations", *Computational Polymer Science* **4**, 75-78 (1994).

11. Robert E. Tuzun, Donald W. Noid, and Bobby G. Sumpter, "Dynamics of a laser driven molecular motor", *Nanotechnology* **6**, 52-63 (1995).

12. Robert E. Tuzun, Donald W. Noid, and Bobby G. Sumpter, "The dynamics of molecular bearings", *Nanotechnology* **6**, 64-74 (1995).

13. Robert E. Tuzun, Donald W. Noid, and Bobby G. Sumpter, "Molecular dynamics treatment of torsional interactions accompanied by dissociation", *Macromolecular Theory and Simulations* **4**, 909-920 (1995).

14. Robert E. Tuzun, Donald W. Noid, and Bobby G. Sumpter, "Computation of internal coordinates, derivatives, and gradient expressions: torsion and improper torsion," *Journal of Computational Chemistry,* Vol. 21, 553-561 (2000)

15. Kazuhiko Fukui, Bobby G. Sumpter, Donald W. Noid, Chao Yang, and Robert Tuzun, "Analysis of eigenvalues and eigenvectors of polymer particles: random normal modes," *Computational and Theoretical Polymer Science,* Vol. 11, 191-196 (2001).

16. P. J. Roebber and L. F. Bosart, The contributions of education and experience to forecast skill. Weather and Forecasting, 11, 21-40, 1996.

17. P. J. Roebber and L. F. Bosart, The complex relationship between forecast skill and forecast value: A real-world analysis. Weather and Forecasting, 11, 544 – 559, 1996.

Learning Computational Methods for Partial Differential Equations from the Web

André Jaun[1], Johan Hedin[1], Thomas Johnson[1], Michael Christie[2], Lars-Erik Jonsson[3], Mikael Persson[4], and Laurent Villard[5]

[1] Alfvén Laboratory, Royal Institute of Technology, SE-100 44 Stockholm, Sweden, jaun@fusion.kth.se, Web-page: http://pde.fusion.kth.se
[2] Center for Educational Development, Chalmers, SE 412 96 Göteborg, Sweden
[3] Unit for Pedagogy and Didactics, University, SE 412 96 Göteborg, Sweden
[4] Electromagnetics, Chalmers Institute of Technology, SE 412 96 Göteborg, Sweden,
[5] CRPP, Ecole Polytechnique Fédérale, CH-1015 Lausanne, Switzerland

Abstract. A course has been developed to learn computational methods from the web[1] and has been tested with postgraduate students from remote universities. Short video conferences or video recordings provide an overview and introduce more detailed studies with numerical experiments in Java-powered course notes. This enables every participant to work at his own pace and to develop his intuition for finite differences, finite elements, Fourier, Monte-Carlo and Lagrangian methods. Assignments are carried out in a regular web browser and are compiled into web pages where the students explain with their own words, equations and programs how to derive, implement and run computational schemes. Our experience shows that the technology is rapidly acquired from templates, using practical examples for the advection, diffusion, Black-Scholes, Burger, Korteweg-DeVries and Schrödinger equations.

1 Introduction

Computational methods are part of the problem-solving skills that need to be mastered by professionals working in a quantitative field. At an advanced level, excellent textbooks generally provide a robust mathematical foundation for one specific approach; they however miss the overview and examples, which are necessary at an introductory level to choose the right method and implement a practical solution. Convinced that the Internet technology can be of great value in this context, we created a *problem-based learning* environment where the acquisition of knowledge is motivated by well defined tasks; in parallel, we switched from a *teaching-centered* to a *learning-centered* course where the students explore the material with the focus, order and pace they choose by themselves. This paper describes the learning method we tested 1997-2000 with summer courses involving 10-20 participants geographically dispersed around Stockholm and Göteborg. Together with the educational material, it is our nice human experience and the encouraging results that we would like to share here with you.

[1] http://pde.fusion.kth.se

V.N. Alexandrov et al. (Eds.): ICCS 2001, LNCS 2073, pp. 1170–1175, 2001.
© Springer-Verlag Berlin Heidelberg 2001

2 A Distance Learning Setup

The course begins with an announcement in the schools' mailing lists[2] outlining the subject. A link to the course notes on the web and former student projects enables potential participants to judge if the content is aligned with their target curriculum. This way of proceeding reflects the current trend towards a free market for university courses and is well adapted to offering teaching services outside traditional school boundaries, such as sister universities and private companies.

Every morning during two weeks, the students are encouraged to attend a lecture in one of the video-conference rooms and may download a video recording from the Internet. Both are optional and serve as an introduction to a second active learning phase where the knowledge is acquired with experiments in the Java-powered course notes. Most of the time is spent carrying out the exercises in a regular browser, form where they are automatically compiled into Web pages. Once they are ready, the students submit their solutions electronically for correction to a (human) teacher. Discussion forums exploit the ability of quick learners to answer simple questions from peers and enable the teacher to focus on the problems where his expertise is most useful and precious. A third week is generally necessary to fulfill all the requirements and is well spent in building up a working knowledge in a variety of methods by solving concrete problems.

Being active researchers in theoretical plasma physics, it is clear that we could only devote a limited amount of time to the entire project. How much can be achieved in a total of six months teaching over three years, how large is the technology burden and how useful are the electronic tools in this context was largely unknown. Moreover, is it reasonable to expect students to visit universities and even pay for courses if the notes are readily accessible from the Internet? Our experience shows that those students who have the possibility to attend classes locally or through video-conference still do so for the stimulation and discussions they get from the teacher and peers. Those who cannot strongly value the flexibility of delivering exercises when and from where they like and yet to benefit from a close personal supervision with corrections from a teacher.

3 Classroom or Video Lectures Provide an Overview

Classroom lectures introduce web page equivalents of the course notes [1] and are broadcast by video conference or (RealVideo) recordings to remote participants. The lessons are short (30-40 minutes) to keep the attention of the audience; every now and then, a short quiz stimulates lively discussions locally before the conclusions are shared in a more orderly manner between remote classrooms. The JBONE applet (Java Bed for ONE dimensional problems) is used to test new schemes directly in the web browser; this adds an unprecedented animation and interactivity to the lecture and is extremely valuable when comparing the numerical properties of different time evolution schemes. Menus select the equation

[2] students97@engineering.myschool.edu, phd-students@yourschool.edu

(advection diffusion, Burger's shock waves, Korteweg-De Vries solitons, Black-Scholes options, Schrödinger, etc), the initial condition (box, Gaussian, cosine, soliton, put option, wavepacket) and editable text fields control the parameters (velocity, diffusion, dispersion, time step, etc) directly in the web browser. A mouse click starts the simulation, making it extremely easy and convincing to illustrate for example the linear instability that occurs when the time step gets too large in an explicit finite difference advection scheme, or to show how the more subtle aliasing in spectral methods affects a non linear train of solitons.

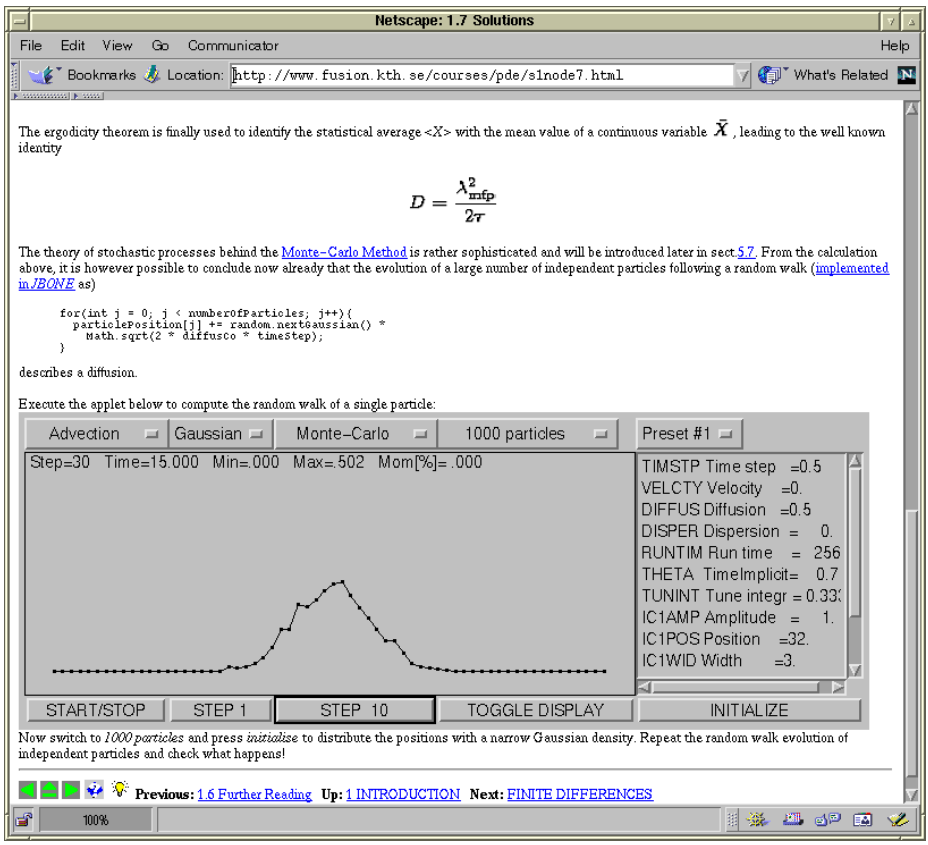


Fig. 1. Screen capture of the web browser displaying an analytical formula, the algorithm with a hyperlink into the source code and the JBONE applet – all after execution of the Monte-Carlo integration with 1000 particles to illustrate the connection between the motion of random walkers and diffusion.

4 Active Learning with Experiments from Home

An advantage of using widespread, platform independent technology is that the students can reproduce and modify the demonstrations back in their office or directly from home. Repeating the line of thought from the classroom, the text and figures take the reader through series of analytical derivations that yield a computational scheme. Hyperlinks point to the relevant sections in the code and show how every algorithm has been implemented. Default parameters are preset to illustrate specific properties, but can be modified to verify if a topic has been correctly understood.

Example: From Brownian motion to diffusion. Both are fundamental in science and engineering and are often hard to understand for undergraduates. An analytical derivation of the RMS displacement $\sqrt{< \xi_i^2 >}$ in a particle's random walk connecting the diffusion coefficient to the square of the mean free path divided by the collision time $D = \lambda_{mfp}^2/2\tau_c$ might en-light a few, but is likely to loose a majority in the algebra. Using the Monte-Carlo evolution from the applet displayed in figure 1, it is simple to demonstrate first how the random motion of a single particle can be described with the Java code

```
for (int j=0; j<numberOfParticles; j++){
  particlePosition[j] +=
     random.nextGaussian() *
     Math.sqrt(2 * diffusCo * timeStep);
}
```

Distributing 1000 particles to approximate an initial Gaussian distribution, the same algorithm immediately shows what diffusion means. Change the initial condition to a cosine or a square box and it is equally rewarding to demonstrate how short wavelength modes decay more rapidly than long wavelengths.

Having part of the students studying away from the campus, is it possible to use the technology to stimulate personal interactions with the teacher and peers? Yes, better than we thought!

5 Assignments Carried out Directly on the Web

Exercises in the first session are designed to familiarize the students with the electronic publishing on the web; templates show how the building blocks are used in relevant schemes to assimilate TEX and Java directly through the context. Teaching at an advanced level where copying is not an issue, we distribute a list of all the solutions web pages and let the students compare and discuss the results

with each other. Our *top pick* of the best solution creates a healthy competition where everybody tries to become a member of a very exclusive list.

Part of the students choose to carry out an additional one week project, applying their favorite method to a topic of interest such as the Black-Scholes equation for a European call option, a tunable finite elements integration for the Schrödinger equation, a mesh refinement procedure, iterative solvers, etc. Given the small amount of time allocated for each project, the scope remains of course limited; by cross-checking each other's reports on the web, the students nevertheless get an overview of a rather broad range of applications.

The material. A single TEXsource generates both the printed course notes and the hyperlinked web pages. Running open software translators such as latex2html [2], tth [3] and scripts embedded in a makefile, the static web material can effectively be produced with no additional cost to what is required to print the notes and slides. Writing the JBONE applet from scratch was quite an effort for the teachers, but the object-oriented language and the encapsulated structure of the code enables students with little programming experience to gradually modify existing schemes and add their own. A substantial amount of documentation (programming tree, keyword index) is created automatically using the javadoc utility, which is part of the standard Java development kit. An automatic download service has been set up for teachers and individuals who would like to use, modify and tailor our material for their specific needs.

6 Working with Discussion Forums

News groups or discussion forums prove to be an ideal tool allowing a sufficient number of participants to interact in a geographically distributed environment. Not only do students help and discuss with each other at virtually any time of the day and night, but the advice they get or provide is usually helpful and competent. Some supervision is required, but instead of answering a dozen times to the same question (often for organizational matters), the teacher can intervene once for an announcement and spend the rest of the time clarifying discussions that remain very informal and help the students to understand the subject in their own language. To encourage interactions between students and create the feeling of belonging to the same virtual classroom, we now reward relevant contributions as well as the assignments.

7 Evaluation and Conclusion

Three classes with 11,18,15 participants went through considerably different learning schemes since the first time the course was taught in a conventional manner (in 1997), introducing a problem based learning with the JBONE applet (1999), using video-conferences between two universities (2000) and allowing now for a distance learning at any time and from anywhere on the Web. We

cannot really draw statistically significant conclusions about the effectiveness of each; anonymous evaluations and discussions with the students do nevertheless indicate that the largest benefits come the *problem-based learning* and the simple user interface. These are much more challenging to implement than hyper-linked documents, when firewalls, different computer platforms, versions of software and operating systems can quickly become a technology nightmare!

Templates provide the most efficient help for the electronic submission of the home assignments. This allows the questions asked in the discussion forums to deal mainly the computational aspects – the substance of the course. One participant proposed to create a discussion group reserved for alumni to maintain valuable contacts after graduation.

In its full electronic form, the course clearly requires a well-maintained web server, which is generally administered by an assistant in a university. Because of the *ask once, answer to all* nature of the discussion forums, assistants can however be employed very efficiently and the overall teaching load is finally similar to a conventional setup.

Some flexibility is required from both the lecturer and the students in order to exploit the new possibilities and work around the weaknesses of a course taught at a distance. Our experience however shows that the pedagogical content is by no means reduced if the technology can be used to support a *problem-based learning* context with a forum allowing the students to discuss and understand the material with their own words. The enthusiasm from all the participants including the teaching assistants is a very gratifying experience and should be an additional encouragement to try similar experiments elsewhere.

Acknowledgements

This work is supported in part by the Summer University of Southern Stockholm (SUSS) and Ericsson.

References

1. Jaun A., Hedin J., Johnson T., Numerical Methods for Partial Differential Equations. TRITA-ALF-1999-05 (1999), `http://pde.fusion.kth.se`
2. Drakos N., Text to Hypertext conversion with LaTeX2HTML. Baskerville **3** (1993) 12, `http://cbl.leeds.ac.uk/nikos/tex2html/doc/latex2html/latex2html.html`
3. Hutchinson, I., TTH: a TeX to HTML translator. `http://hutchinson.belmont.ma.us/tth/`

Computational Engineering and Science Program at the University of Utah

Carleton DeTar[3], Aaron L. Fogelson[2], Christopher R. Johnson[1], Christopher A. Sikorski[1]

[1]School of Computing, {crj, sikorski}@cs.utah.edu
[2]Department of Mathematics, fogelson@math.utah.edu
[3]Department of Physics, detar@physics.utah.edu
University of Utah, Salt Lake City, Utah 84112.

Abstract

We describe a new graduate program at the Univesity of Utah that we consider a first step towards the modernization of the University's curriculum in what we call "Computational Engineering and Science" (CES). Our goal is to provide a mechanism by which a graduate student can obtain integrated expertise and skills in all areas that are required for the solution of a particular computational problem.

1 Computational Engineering and Science Program

The grand computational challenges in engineering and science require for their resolution a new scientific approach. As one report points out, "The use of modern computers in scientific and engineering research and development over the last three decades has led to the inescapable conclusion that a third branch of scientific methodology has been created. It is now widely acknowledged that, along with traditional experimental and theoretical methodologies, advanced work in all areas of science and technology has come to rely critically on the computational approach." This methodology represents a new intellectual paradigm for scientific exploration and visualization of scientific phenomena. It permits a new approach to the solution of problems that were previously inaccessible.

At present, too few researchers have the training and expertise necessary to utilize fully the opportunities presented by this new methodology; and, more importantly, traditional educational programs do not adequately prepare students to take advantage of these opportunities.

Too often we have highly trained computer scientists whose knowledge about engineering and sciences is at the college sophomore, or lower, level. Traditional educational programs in each of these areas stop at the sophomore level — or earlier — in the other area. Also, education tends to be ad hoc, on the job and self-taught."

V.N. Alexandrov et al. (Eds.): ICCS 2001, LNCS 2073, pp. 1176–1185, 2001.
© Springer-Verlag Berlin Heidelberg 2001

This situation has arisen because the proper utilization of the new methodology requires expertise and skills in several areas that are considered disparate in traditional educational programs. The obvious remedy is to create new programs that do provide integrated training in the relevant areas of science, mathematics, technology, and algorithms. The obvious obstacles are territorial nature of established academic units, entrenched academic curricula, and a lack of resources.

At the University of Utah the School of Computing (located in the College of Engineering), with the Departments of Mathematics and Physics (located in the College of Science) have established a graduate program that we consider a first step towards the modernization of the University's curriculum in what we call "Computational Engineering and Science" (CES). Our goal is to provide a mechanism by which a graduate student can obtain integrated expertise and skills in all areas that are required for the solution of a particular problem via the computational methodology.

We have built upon an established certificate program and recently created an M.S. CES degree program. If the M.S. CES program is successfel, we will consider expanding the program to a Ph.D. in CES. Our program is designed mostly for students in the Colleges of Engineering, Mines, and Science. However, in principle any graduate student at the University can participate.

To obtain the MS degree in CES, a student must complete courses and present original thesis research (for the thesis option) in each of the following areas:

I. Introduction to Scientific Computing

II. Advanced Scientific Computation

III. Scientific Visualization

IV. Mathematical Modeling

V. Case Studies in CES

VI. Elective course

VII. Seminar in Computational Engineering and Science

Of the above items, only V and VII are truly new requirements. Numerical Analysis has been taught in our departments for several years. Mathematical modeling has been spread over a large number of courses in the current Mathematics curriculum; the new course has been designed particularly for the CES program and has replaced one or more other courses in the students' load. The situation in regard to courses I and III in the Computer Science curriculum is very similar.

All courses are designed for first- or second-year science and engineering graduate students who have a knowledge of basic mathematics and computing skills. The most innovative aspect of our CES program is course V., Case Studies in CES. This course consists of presentations by science and engineering faculty from various departments around the campus. These faculty, all active in computationally intensive research, introduce students to their own work over intervals of, typically, three weeks. The course provides students with a reasonably deep understanding of both the underlying science and engineering principles involved in the various projects and the practical issues confronting the researchers. It also provide a meeting place for faculty and graduate students engaged in CES activities in various departments throughout campus. In the CES seminar (VII) students are be required to report on their own CES activities to their peers.

This approach, then, will serve the students by at once focusing their activities and bringing together in one place several essential components of CES that were previously spread over a larger and less clearly defined part of the existing curriculum. In addition, the program will help students learn to work with researchers in other disciplines and to understand how expertise in another field can help propel their own research forward.

The program is administered by a director and a "steering committee" consisting of two members each from the departments of mathematics, physics and the school of computing. The committee is advised by a board of faculty active in CES research.

2 Computational Engineering and Science Research

The University of Utah has a rich pool of faculty who are active in the various areas involved in CES. These include fields such as computational fluid dynamics, physics and chemistry, earthquake simulation, computational medicine, pharmacy, biology, computational combustion, materials science, climate modeling, genetics, scientific visualization and numerical techniques. A few examples of some of the current research in these areas follow.

2.1 Scientific visualization

Common to many of the computational science application areas is the need to visualize model geometry and simulation results. The School of Computing at the University of Utah has been a pioneer and leader in computer graphics and scientific visualization research and education. Some of the first scientific visualizations were invented and displayed here (e.g. , the use of color for finite element analysis and the use of geometrical visualizations of molecules). Furthermore, is one of the sites of the NSF Science

and Technology Center (STC) for *Computer Graphics and Scientific Visualization*. The STC supports collaboration on a national level among leading computer graphics and visualization groups (Brown, CalTech, Cornell, UNC, and Utah) and provides graduate students with a unique opportunity for interaction with this extended graphics family. Utah is also home to the Scientific Computing and Imaging (SCI) Institute [1]. SCI Institute researchers have innovated several new techniques to effectively visualize large-scale computational fields [2, 3, 4, 5, 6, 7, 8, 9, 10, 11].

Example of SCI Institute visualization work is shown in Figures 1-3.

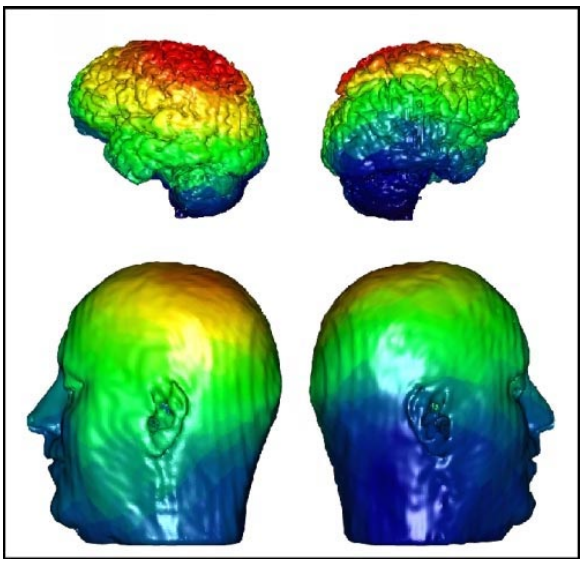


Figure 1: Inverse EEG Simulation.

2.2 Computational Combustion

Professor Philip Smith of the Department of Chemical Engineering has been developing large scale computer simulators that couple 3D computational fluid dynamics with reaction, heat transfer, radiation, turbulence, and particle transport to compute local combustion behavior in full scale utility boilers. Professor Smith has been collaborating with Dr. Kwan-Liu Ma and Professor Chris Sikorski of the Department of Computer Science to develop computer visualization methods for better understanding the simulation results. Their research contributes to the technological advances of the United States by helping to minimize pollutant formation and to maximize efficiency for combustion systems.

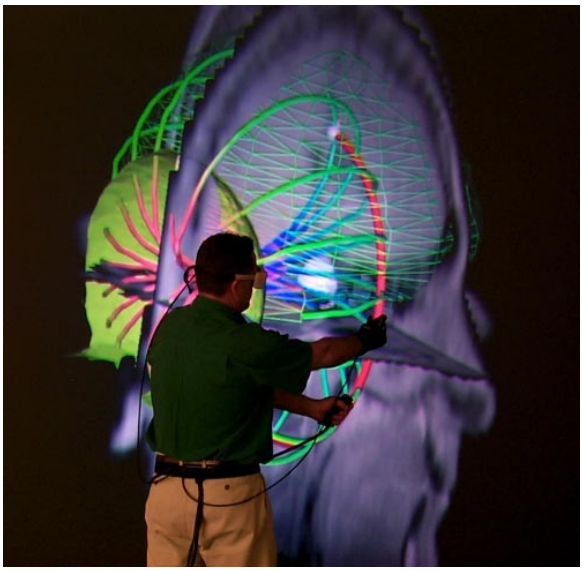


Figure 2: Dr. Greg Jones, Associate Director of the SCI Institute, interacting with a large-scale model of a patient's head within a stereo, immersive environment. The colored streamlines indicate the current from a simulation of epilepsy.

The University of Utah has created an alliance with the DOE Accelerated Strategic Computing Initiative (ASCI) to form the Center for the Simulation of Accidental Fires and Explosions (C-SAFE). It focuses specifically on providing state-of-the-art, science-based tools for the numerical simulation of accidental fires and explosions, especially within the context of handling and storage of highly flammable materials. The objective of C-SAFE is to provide a system comprising a problem-solving environment in which fundamental chemistry and engineering physics are fully coupled with non-linear solvers, optimization, computational steering, visualization and experimental data verification. The availability of simulations using this system will help to better evaluate the risks and safety issues associated with fires and explosions. Our team will integrate and deliver a system that will be validated and documented for practical application to accidents involving both hydrocarbon and energetic materials.

2.3 Mathematical and Computational Biology

The Mathematics Department at the University of Utah is home to one of the world's largest and most active research groups in Mathematical Biology. The work in this group centers on understanding the biological mechanisms that regulate the dynamics of important physiological, biochemical, biophysical, and ecological interactions. A

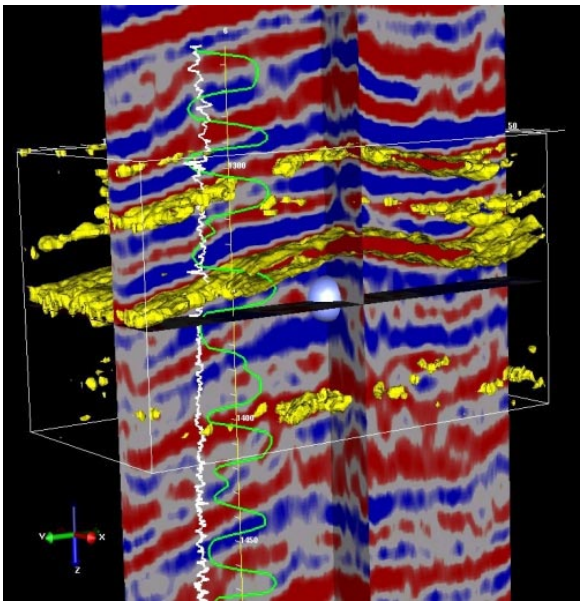


Figure 3: Visualization of seismic data.

major focus of research, lead by Aaron Fogelson, is in using mathematics and computation to understand how the complex biochemical and biophysical components, especially the fluid dynamics, of platelet aggregation and coagulation interact in hemostasis (normal blood clotting) and thrombosis (pathological blood clotting within blood vessels). This fascinating area has tremendous practical importance because thrombosis is the immediate cause of most heart attacks and strokes. Because the models of clotting are very complex (they involve fluid dynamics, fluid-structure interactions, chemical kinetics, chemical and mass transport), they pose substantial computational challenges, and have required the development of novel numerical methods and software to meet these challenges. This software has been and is being applied to a wide range of biological problems in which fluid flow plays an important role. A second major research area, led by James Keener, involves modeling and three-dimensional computation of electrical waves in cardiac muscle. The goal is to understand normal signal propagation and the coupling between the electrical stimuli and cardiac muscle contraction, to understand the mechanisms underlying the onset of pathological arrythmias, and to understand at the cellular and tissue level how defibrillation works so as to help optimize defibrillation strategies. Some of the other work in the group includes studies of information processing in the primary visual cortex, models of territoriality in interacting animal populations, and studies of invasion of ecosystems by new species.

2.4 Computational Medicine

An interdisciplinary team of nationally recognized research centers at Utah involving the Cardiovascular Research and Training Institute (CVRTI), the Scientific Computing and Imaging (SCI) Institute, the Center for Advanced Medical Technology (CAMT), and the Neurosurgery Department are working together to tackle large-scale computational problems in medicine.

Every year, approximately 500,000 people die suddenly because of abnormalities in their hearts' electrical system (cardiac arrhythmias) and/or from coronary artery disease. While external defibrillation units have been in use for some time, their use is limited because it takes such a short time for a heart attack victim to die from insufficient oxygen to the brain. Lately, research has been initiated to find a practical way of implanting electrodes within the body to defibrillate a person automatically upon onset of cardiac fibrillation. Because of the complex geometry and inhomogeneous nature of the human thorax and the lack of sophisticated thorax models, most past design work on defibrillation devices has relied on animal studies. We have constructed a large-scale model of the human thorax, the Utah Torso Model [12, 13, 14, 15], for simulating both the endogenous fields of the heart and applied current sources (defibrillation devices). Using these computer models, We are also able to simulate the multitude of electrode configurations, electrode sizes, and magnitudes of defibrillation shocks. Given the large number of possible external and internal electrode sites, magnitudes, and configurations, it is a daunting problem to computationally test and verify various configurations. For each new configuration tested, geometries, mesh discretization levels, and a number of other parameters must be changed.

Excitation currents in the brain produce an electrical field that can be detected as small voltages on the scalp. By measuring changes in the patterns of the scalp's electrical activity, physicians can detect some forms of neurological disorders. Electroencephalograms, EEGs, measure these voltages; however, they provide physicians with only a snapshot of brain activity. These glimpses help doctors spot disorders but are sometimes insufficient for diagnosing them. For the latter, doctors turn to other techniques; in rare cases, to investigative surgery.

Such is the case with some forms of epilepsy. To determine whether a patient who is not responding to medication has an operable form of the disorder, known as temporal lobe epilepsy, neurosurgeons use an inverse procedure to identify whether the abnormal electrical activity is highly localized (thus operable) or diffused over the entire brain.

To solve these two bioelectric field problems in medicine, we have created two problem solving environments, SCIRun and BioPSE [16, 17], to design internal defibrillator devices and measure their effectiveness in an interactive graphical environment

[18]. Similarly we are using these PSEs to develop and test computational models of epilepsy [19] as shown in Figure 1.

Using SCIRun and BioPSE, scientists and engineers are able to design internal defibrillation devices and source models for the epileptic foci, place them directly into the computer model, and automatically change parameters (size, shape and number of electrodes) and source terms (position and magnitude of voltage and current sources) as well as the mesh discretization level needed for an accurate finite element solution. Furthermore, engineers can use the interactive visualization capabilities to visually gauge the effectiveness of their designs and simulations in terms of distribution of electrical current flow and density maps of current distribution.

2.5 Computational Physics

Several computational science opportunities are offered by the Physics Department. Students can gain experience in computational physics on platforms ranging from small Beowulf clusters to supercomputers at the national laboratories to the most powerful, special-purpose computers available in the world.

1. High Energy Astrophysics

   The High Resolution Fly's Eye and the High Energy Gamma Ray research groups collect terabytes of observational data that require processing on high performance parallel computers. Monte Carlo simulations of detector performance and of theories also require intensive computation. The goal of this research is to understand the origins and production mechanisms of the mysterious, energetic particles. Professors Pierre Sokolsky, David Kieda, Eugene Loh, Charles Jui, Kai Martens, Wayne Springer, and Vladimir Vassiliev participate in this work.

2. The Evolution of the Universe

   Computer models provide unique insights into the evolution of galaxies, the formation of planets in stellar nebulae, and the analysis of the structure of the early universe. These are some of the research interests of Professor Benjamin Bromley.

3. Solving the Strong Interactions

   Large scale *ab initio* simulations of quantum chromodynamics, the theory of interacting quarks and gluons, have proven to be an indispensible guide to our understanding of the masses and structure of the light elementary particles, the decays of heavy mesons, and the cooling of the quark-gluon plasma, which existed in the early universe. Professor Carleton DeTar is carrying out this work.

2.6 Future Directions

The field of Computational Engineering and Science holds rich possibilities for future development. The *computational paradigm* has taken hold in nearly every area in science and engineering. Its use is also becoming more common in many fields outside of science and engineering, such as the social sciences, architecture, business, and history. Its success hinges on researchers' ability and willingness to transcend traditional disciplinary barriers and share expertise and experience with a large group of colleagues who may have been perceived previously as working in unrelated fields. Such boundary-crossing enriches researchers' work by providing new computational opportunities and insights. Building the interdisciplinary spirit will enable the solution of problems that were previously inaccessible. At the University of Utah, we are excited by such prospects and have taken the first step in initiating a mechanism to educate the next generation of scientists and engineers.

For more information on our CES program visit www.ces.utah.edu.

References

[1] Scientific Computing and Imaging Institute: www.sci.utah.edu.

[2] C.R. Johnson, Y. Livnat, L. Zhukov, D. Hart, and G. Kindlmann. Computational field visualization. In B. Engquist and W. Schmid, editors, *Mathematics Unlimited – 2001 and Beyond*. 2001 (to appear).

[3] H.W. Shen and C.R. Johnson. Differential volume rendering: A fast alogrithm for scalar field animation. In *Visualization 94*, pages 180–187. IEEE Press, 1994.

[4] H.W. Shen and C.R. Johnson. Sweeping simplices: A fast isosurface extraction algorithm for unstructured grids. In *Visualization '95*. IEEE Press, 1995.

[5] H.W. Shen, C.R. Johnson, and K.L. Ma. Global and local vector field visualization using enhanced line integral convolution. In *Symposium on Volume Visualization*, pages 63–70. IEEE Press, 1996.

[6] C.R. Johnson, S.G. Parker, C. Hansen, G.L. Kindlmann, and Y. Livnat. Interactive simulation and visualization. *IEEE Computer*, 32(12):59–65, 1999.

[7] Y Livnat, H. Shen, and C. R. Johnson. A near optimal isosurface extraction algorithm using the span space. *IEEE Trans. Vis. Comp. Graphics*, 2(1):73–84, 1996.

[8] Y. Livnat and C.D. Hansen. View dependent isosurface extraction. In *IEEE Visualization '98*, pages 175–180. IEEE Computer Society, October 1998.

[9] Y. Livnat, S.G. Parker, and C.R. Johnson. Fast isosurface extraction methods for large image data sets. In A.N. Bankman, editor, *Handbook of Medical Imaging*, pages 731–745. Academic Press, San Diego, CA, 2000.

[10] G.L. Kindlmann and D.M. Weinstein. Hue-balls and lit-tensors for direct volume rendering of diffusion tensor fields. In *Proceedings of the IEEE Visualization 99*, pages 183–189, 1999.

[11] G.L. Kindlmann and D.M. Weinstein. Strategies for direct volume rendering of diffusion tensor fields. *IEEE Trans. Visualization and Computer Graphics*, 6(2):124–138, 2000.

[12] C.R. Johnson, R.S. MacLeod, and P.R. Ershler. A computer model for the study of electrical current flow in the human thorax. *Computers in Biology and Medicine*, 22(3):305–323, 1992.

[13] C.R. Johnson, R.S. MacLeod, and M.A. Matheson. Computational medicine: Bioelectric field problems. *IEEE COMPUTER*, pages 59–67, October 1993.

[14] R.S. MacLeod, C.R. Johnson, and M.A. Matheson. Visualization tools for computational electrocardiography. In *Visualization in Biomedical Computing*, pages 433–444, 1992.

[15] R.S. MacLeod, C.R. Johnson, and M.A. Matheson. Visualization of cardiac bioelectricity — a case study. In *IEEE Visualization '92*, pages 411–418, 1992.

[16] C.R. Johnson and S.G. Parker. Applications in computational medicine using SCIRun: A computational steering programming environment. In H.W. Meuer, editor, *Supercomputer '95*, pages 2–19. Springer-Verlag, 1995.

[17] S.G. Parker, D.M. Weinstein, and C.R. Johnson. The SCIRun computational steering software system. In E. Arge, A.M. Bruaset, and H.P. Langtangen, editors, *Modern Software Tools in Scientific Computing*, pages 1–44. Birkhauser Press, 1997.

[18] J.A. Schmidt, C.R. Johnson, and R.S. MacLeod. An interactive computer model for defibrillation device design. In *International Congress on Electrocardiology*, pages 160–161. ICE, 1995.

[19] D.M. Weinstein, L. Zhukov, and C.R. Johnson. Lead-field bases for EEG source imaging. *Annals of Biomedical Engineering*, 28:1–7, 2000.

Influences on the Solution Process for Large, Numeric-Intensive Automotive Simulations

Myron Ginsberg, Ph.D.

HPC Research and Education
Farmington Hills, Michigan 48335-1222 USA
m.ginsberg@ieee.org

Abstract. This paper focuses on the performance limitations of solving large automotive simulation applications. Some recommendations are given for improving the solution process. The content is based upon the author's experience in the U.S. automotive industry and on his recent investigation of U.S. automotive applications while he was a NASA/ASEE Summer Research Fellow in the Computational AeroSciences Team at NASA Langley Research Center. Most of the applications are in the following areas: computational fluid dynamics (CFD), crash analysis, Noise-Vibration-Harshness (NVH) modeling, manufacturing problems such as metal forming processes, and multidisciplinary optimization problems such as combinations of NVH and crash analysis.

1 Introduction

The global competitiveness of each U.S. automotive company is dependent upon how effectively it can continually reduce lead time between concept and production of new vehicles. By doing so, the individual auto companies can rapidly respond to utilization of new technologies, shifting customer needs and desires as well as competition from both European and Asian rivals. In 1980, the lead time between concept and production was 60 months. In early 2001 it is between 20 and 24 months with an industry target of 18 months or less. To meet such a goal, several changes are required: (1) The amount of physical prototyping must continue to be reduced and replaced by realistic three-dimensional, math-based/physics-based computer modeling; (2) Such computer models must employ very aggressive use of modest and massively parallel hardware, software, and algorithm technologies; (3) To accomplish (1) and (2) above requires extensive multidisciplinary interactions and sharing of resources with industrial/government consortia beyond the automotive companies in which the problems arise; (4) Success requires access to new computer architectural resources as well as hardware, software, and algorithms to experiment with optimal mapping of those resources to produce the fastest, most accurate, and economical solutions.

In the following sections we discuss features of the automotive computational environment (Section 2), properties of the large-scale automotive applications

V.N. Alexandrov et al. (Eds.): ICCS 2001, LNCS 2073, pp. 1189-1198, 2001.
© Springer-Verlag Berlin Heidelberg 2001

investigated (Section 3), roadblocks to a fast solution (Section 4), approaches to reducing computer solution times (Section 5), recommendations to improve performance (Section 6), summary and conclusions (Section 7). Additional details about the specific problems considered and the nature of the investigation as well as the motivation for the study can be found in [1] and [2], respectively.

2 Automotive Computational Environment

Until the mid 1980s, most automotive companies depended heavily on in-house developed proprietary software for solving their scientific and engineering problems. As the computer technology began to rapidly evolve and the problems became larger and more complex, it became quite apparent that it would be much more cost effective to outsource the development of such software. The net effect was the creation of many commercial independent software vendors (ISVs). At present, approximately a dozen major ISVs serve the growing needs of the automotive industry. These ISVs are generally relatively small companies which primarily focus on supporting one or more application codes that can be used to solve automotive problems on several computer platforms. Most of these codes have been written in FORTRAN and C but in recent years there has been a slow movement to utilize C++ and consider other object-oriented languages such as Java [3].

Global competition has forced the auto companies to drastically reduce their lead times between vehicle concept and production. Lead time reduction has been accomplished by a rapid transition from physical prototyping to math-based computer modeling and simulation. This in turn has required the need for accurate, realistic, and fast computer simulations of automotive problems, often requiring very aggressive use of parallel computing approaches.

The typical large-scale automotive problems are in four categories: (1) crashworthiness modeling; (2) noise, vibration, and harshness (NVH) applications; (3) computational fluid dynamics which includes both exterior aerodynamics and combustion modeling; (4) manufacturing processes such as metal forming applications. More details about all four categories can be found in [4], [5], [6], [7].

The computers for solving automotive scientific / engineering applications in the 1970s and early 1980s were usually the same traditional mainframes used for large data processing problems. Then in early 1984 General Motors Research acquired the first in-house Cray supercomputer in the world automotive industry [7]; the trend during the next seven years was for most of the world automotive companies to acquire their own Cray vector supercomputers as well as some of the Japanese supercomputers from NEC, Fujitsu, and Hitachi. From the mid 1990s until the present, the trend has been to acquire scalar concurrent parallel machines such as the IBM SP, SGI O2000 and O3000 series, HP9000 series, and Sun Enterprise 10000 systems. Many of the European and Japanese auto companies continue to use vector supercomputers especially those from the aforementioned Japanese vendors. In the U.S. most recently, only Ford Motor Company has continued to use several vector supercomputers [8].

3 Properties of the Large-Scale Automotive Problems

The author collected large-scale automotive applications submitted from General Motors, Ford Motor Company, and DaimlerChrysler. Details about this investigation as well as descriptions of all eight problems can be found in a forthcoming paper [1]. The investigation focused on identifying and documenting a diverse collection of large-scale, very important, compute-intensive automotive industry problems. At present, most of these problems cannot readily be solved in a commercial industrial environment because of computer limitations and/or memory or I/O requirements. The intent of this study was to follow up with many of these applications to determine if these problems could be economically solved by utilizing leading-edge hardware, software, and/or algorithm technologies that are not readily accessible within the auto companies. Below is a list of the eight problems submitted; see [1] for details about each of these applications.

- Transient analysis of a V6 exhaust manifold using a coupled 1D/3D Model
- Solving normal modes for models with 100,000 to 500,000 grids, approximately 600,000 to 3,000,000 Degrees of Freedom (DOF)
- Simulation of vehicle impact in real-world crashes
- Metal forming problem involving solution of 500,000 linear simultaneous equations, 1000 times in the entire analysis
- Establish statistical characteristics of vehicle road noise performance using the Monte Carlo simulation method
- Full vehicle external flow simulation
- Stochastic multidisciplinary shape optimization of crash and NVH
- Mid frequency vibration problems

There was no attempt to solve these problems due to the limited time available during the summer initial investigation; however, observations were made about the nature of the bottlenecks that prevented efficient solutions. Those will be discussed in the next section of this paper.

There were several observations that can be made about these problems. Although these applications were submitted in isolation from one another by three different auto companies, there was a remarkable similarity about the size and nature of all problems submitted. The increasing pressure on the auto companies to reduce lead time is forcing them not only to drastically increase the amount of computer simulation modeling but also to do so at an increasingly faster pace. The ultimate goal of automakers is to be able to create computer models of all vehicle attributes including realistic modeling of the possible interactions amongst subsystems. With such a capability, modifications can be easily accomplished in early design phases and much more economically than at times closer to actual production. To be successful requires fast turnaround of computer simulations of any subsystem and this usually means at least one complete detailed simulation per day and/or via overnight turnaround. None of the eight problems listed above are being solved that quickly at present. Although all these problems are now being solved in a parallel computing environment, much

more aggressive and innovative approaches are needed to significantly speed up the solution process within the auto companies.

4 Roadblocks to a Fast Solution

The comments made in this section are not intended to place blame on any entity for impeding progress in solving these large problems efficaciously. Often times it has been too risky financially to make some of the needed changes and / or to do so would be too time-consuming in an industrial environment.

Several of the problems listed in Section 3 could indeed most likely be solved today much faster than they are currently being solved within the auto companies. Let's look at some of the impediments. As indicated in Section 2 the auto companies have become almost totally dependent on the ISV-based software that they all use to solve their large applications. This is both an economic blessing and a curse. It saves the auto companies the money they would have to spend in-house to develop the application software but at the same time it excludes immediate use of existing software that could solve the problem much faster than at present. All of the automotive companies are very reluctant to use a computer solution external to their ISV-based software even if the solution is faster and/or more accurate. Since these ISV-based companies are relatively small compared to the auto companies they serve, some do not have the people and computer resources nor the time to conduct extensive research of new and better algorithms or to perform extensive code rewrites of their existing software. The ISVs often need to prioritize in order to respond to the demands of a diverse customer base. Furthermore, when new computer hardware technology becomes available, the ISVs are understandably reticent to port to the new platforms because of the economics of doing so and not readily knowing how rapidly their customers will move to the new architecture. The net result is delays in using new algorithms and architectures to attack these large problems.

There is also a reluctance mostly on the part of the auto companies to port to new operating systems. Thus, for example, although several ISV-based software packages [9] have been ported to a Linux environment, this has not happened at auto companies.

Another impediment to computational improvement is the slow creation and integration of new algorithms into the ISV-based software. Because the ISVs do not have the manpower nor the time to research and create new algorithms and extensively test them, they must of necessity rely on other sources such as universities and/or government labs. Some ISVs have very strong links with academia but there still persist problems with rapid implementation of new academic algorithms in production versions of ISV-based codes. Traditionally, only industry pressure (and sometimes dollars) hastens the process.

Another related difficulty is that there are some inherent problems in algorithm creation that have yet to be resolved before accurate and totally realistic automotive simulations can be achieved. For example, it is very difficult to implement accurate

error propagation analysis and do it reasonably fast. Secondly, in some cases the physics of the physical component interactions is not yet completely understood, thus potentially producing deceptive results in the computer simulation. Also the need to deal with multiphysics phenomena complicates the needed algorithm but must be dealt with successfully to promote realistic simulations.

An additional barrier is limited access to a wide variety of computer platforms that might provide faster execution on a specific automotive application than achieved by an auto company on its in-house computers. This situation is further compounded by the fact that a potentially faster machine might be inaccessible to an auto company or not yet even have a port of the ISV-based software that the automaker utilizes to solve such problems. In addition, it may be very difficult and/or time consuming to determine if indeed the external architecture would produce a better and faster results than on the existing in-house machine. This latter difficulty is exacerbated by the lack of any relevant existing hardware performance comparison data.

All the above-mentioned dilemmas are further aggravated by the lack of any significant software, algorithm exploration or hardware investigations by the in-house automotive people. They are rightfully focused on the vehicle engineering problems, not the software to provide accurate and fast simulations. Thus the burden generally falls unfairly on the ISVs but they too lack the resources to do extensive hardware performance comparisons of their software, especially as new platforms become available. Many of the issues described above fall into the cracks amongst the hardware vendors, the ISVs, the academic algorithm creators, and the auto companies. There obviously needs to be more interaction addressing these problems and day-to-day communications amongst these entities.

5 Some Current Activities to Improve Computer Solution

Despite the difficulties described in the previous section, there have been several recent activities which have significantly reduced computation time for these large-scale automotive problems. We present a few examples here.

Dr. Sobieszczanski-Sobieski (Leader of the Computational AeroSciences Team) of the NASA Langley Research Center was able to substantially reduce total wall clock solution time for an earlier problem from Ford Motor Company; details are given in [10]. Essentially this was an optimization of a car body for crashworthiness, NVH, and weight; this problem was formulated as a stochastic multidisciplinary shape optimization of crash and NVH using two automotive ISV-based codes (MSC's NASTRAN and Mecalog's RADIOSS). The goal is to obtain peak performance while design parameters (geometry shape, spot weld pattern, thickness, and material properties) are modified and at the same time reduce the variation of that performance; this is necessary because the design process based on nominal values is not adequate due to physical variation in manufacturing and material which can significantly impact the actual products. The problem attributes are: NVH model is a BIW-trimmed body structure without powertrain and suspension subsystems and using

MSC NASTRAN finite element model of 350,000 + Degrees of Freedom, normal modes, static stress and design sensitivity analysis using Solution Sequence 200 (in version 70.5); there are 29 design variables (sizing, spring stiffness); total computer time was dominated by the crash model (run using RADIOSS, version 4.1b); both NVH and crash models were executed on an SGI O2000.

Fine grain parallelism helped reduce optimization procedure total elapsed time of 292 hours to 24 hours for a single analysis using 12 processors on an SGI O2000. Coarse grain parallel computing for this problem also provided significant reduction in total elapsed time. The total net effect for this problem was a reduction from 257 days of elapsed computing time to 1 day. This was achieved by utilizing the MDO approach in a parallel computing environment without altering any of the NASTRAN or RADIOSS source code. This approach would also be effective for use on one or more of the problems referred to in Section 3.

There are several additional improvements that could be beneficial to the automotive simulation process but space limitation here prevents further discussion. The interested reader should look at [11], [12], and/or the references cited in the rest of this section. Many recent developments are reported in HPCWIRE (http://www.hpcwire.com) and references to specific articles are listed below.

In the visualization area, the following activities could positively stimulate further automotive simulation improvements. Virtual reality is being used to "simulate the entire process of automotive engine constituting from designing, manufacturing including the motion of the engine" [13]; this virtual engine factory will be developed using ESI software on a NEC SX supercomputer for a Japanese auto company. Penn State University and ESI are developing a system that makes it possible to do "virtual" tire testing; i.e., "a facility to roadtest a tire design virtually while the tire is still on the drawing board." [14] Mazda and MTS System Corp. have a vision of integrated simulation - analytical, physical and virtual." [15] Ford and U.K's Defence Evaluation Agency are developing "technology to create full-scale virtual digital models of prototype vehicles which gives Ford the ability to re-design features in real-time and the capability to hold multiple design reviews simultaneously." [16] There are efforts to combine tele-immersion and virtual reality "to create a 3-D visual environment similar to that depicted by the Holodeck on the Star Trek TV show." [17] Some interesting leading-edge work with BMW, SGI, and the University of Erlangen involves the VTCrash virtual reality software written in C++, implemented on SGI Onyx and Octane workstations and which allows computational steering within the model where the average model consists of 200,000 finite elements and the scene visualization "uses one processor of an Infinite Reality Onyx with at least 10 frames per second." [18]

In the HPC architecture area there are several developments which will directly stimulate automotive simulation. Cray just announced plans for a mid-2001 release of an Alpha-based Linux supercluster system in response to customer requests for a "re-architected T3E using leading off-the shelf cluster technologies." [19] "Beowulf class machines using commodity chips are economical machines for some HPC applications but not as capability efficient in throughput for conventional large-scale applications." [20] So far, they have not been utilized in the auto industry. Vector class

supercomputers remain widely used in Europe and Japan and at Ford in the U.S. where they are used for "safety and structural analysis."[8] It is interesting to note in Germany that Fujitsu, NEC, Hitachi, and Cray (both T3E and SV1) supercomputers are used but "commercial users do not install the ultra high-end machines" but often share HPC hardware resources with academia; for example, at University of Stuttgart, 50% of HPC resources are shared by U of Karlsruhe and U of Stuttgart, 40% is used by Debis Systemhaus (subsidiary of DaimlerChrysler) and 10% is used by Porsche." [21] A new effort lead by IDC and involving industry and government HPC representatives is attempting to create new effective, realistic HPC performance metrics and benchmarks which could help industry to evaluate HPC architectures [22].

There are several software performance improvement projects which could be beneficial to the auto companies simulation efforts. SGI and ESI using an SGI O3000 with 96 MIPS R12000 processors and ESI PAM CRASH 2000 software achieved a "sustained rate of 12 GFLOPS for a BMW crash model; this was the highest level of performance ever achieved in computer crash simulation." [23] Cray and Mcube are focusing on accurate simulation of aeroacoustics such as "alleviating wind noises from side mirrors and driver's side structures and exhaust system noise." [24] Cray and MSC.Software Corp. have produced a "900% speedup in key calculations for NVH problems which implies massive NVH optimization jobs can now be performed as overnight turnaround rather than requiring several days." [25]

Detailed information about some of the above mentioned performance improvement activities is not readily available. Some views may be overly optimistic, represent targets rather than achievements and/or marketing hyperbole but it is the responsibility of each auto company to separate the wheat from the chaff.

6 Recommendations to Improve Problem Solving

From the remarks in the preceding two sections, it should be apparent that there are four areas in which improvements are needed: (1) increased use of visualization, (2) performance comparisons of HPC architectures, (3) algorithm development and dissemination, and (4) software performance improvements.

The shift from physical prototyping to computer simulation requires tightly integrated use of virtual reality and teleimmersion capabilities with the ISV-based applications software; progress in this area has been very slow. Active collaboration amongst auto companies, the NSF centers, government research labs, and ISVs could greatly enhance this effort. So far activities in this area have been very sporadic. Daily interactions amongst these entities are needed with clear goals to create immediate pragmatic impact on these large-scale problems.

Acquisition of new computer hardware requires thorough scrutiny of existing alternatives; however, the auto companies having to cope with severe time constraints, have often had to make decisions based upon very limited investigations and/or on unsubstantiated vendor claims. A "Consumer's Report" type of ongoing benchmarks should be created to provide objective performance comparisons using ISV-based

software on realistic automotive problems across a variety of computer platforms. The results should be updated on a continual basis and available on the Internet, possibly on a subscription basis to interested auto companies. Such performance information would be helpful to the entire automotive community in making hardware selections. New practical and realistic metrics are needed to measure "real" performance; the recent efforts initiated by IDC [22] seem to be in the right direction although widespread automotive industry involvement so far appears to be minimal.

In the algorithm development area there needs to be a pragmatic focus on tech transfer from government and academic algorithms to the automotive ISV-based codes. For example, one of the problems listed in Section 3 is very likely to be solved much faster using a code available from ICASE [26] but this is unacceptable at present because the code is not part of the ISV-based software the automaker uses. A cooperative effort should be initiated with government and academic algorithm developers directly working with the ISVs to provide fast ports for important applications.

In the software area there is a need for the ISVs to transition their legacy FORTRAN and C based codes to an object-oriented environment such as C++, Java, or even object-oriented FORTRAN. Such a move would in the long run drastically reduce maintenance costs and time as well as help promote use of grid technologies [27] that are now emerging on the Internet and which would be very amenable to use with cluster machines for large applications.

Another software issue which should be addressed is the gap in the creation of the ISV-based software: a computational science gap, i.e., we have experts examining the physics and engineering issues and another group focusing on writing thousands of lines of code but there is no one in the middle to interface new software developments such as VR, grids on the Internet, and user friendly parallel computing tools; most of the work in those areas is being done in a few government labs and academia but transferring that expertise to the ISVs has been very slow. Hopefully, the new breed of computational scientists that are emerging from about two dozen universities in the U.S. and elsewhere will soon have positive impact on ISV progress in this area.

In the visualization area perhaps the automotive people should be seeking closer collaboration with the visualization people in the motion picture industry. Visualization technology (real-time animation and VR) is the cornerstone for successful comprehensive simulation in the automotive industry. An observation at BMW [18] indicates that "30 percent of the efforts involved in a typical simulation go into the preprocessing phase and about 10 percent is taken up by the actual computation, while approximately 60 percent goes into the analysis and communication of the results. Clearly there is a strong incentive to reduce the last percentage through the implementation of insightful, intuitive visualization tools which allow effective communication between engineers." [18] Real-time (or near real-time) computational steering is essential to the total automotive simulation process; faster hardware and improved visualization techniques with large datasets are rapidly making that possible. Such visualization capabilities as demonstrated by activities with BMW [18] need to be rapidly integrated into the automotive ISV-based software.

7 Summary and Conclusions

Conversion costs, performance concerns, time constraints, and desire for a stable computational environment are primary barriers to extensive adoption of new technology for large-scale automotive simulation problems. Improvements in software tools including visualization capabilities, creation of numerical libraries, use of optimizing compilers, and pragmatic cooperative academic/industrial/government interactions can stimulate auto industry movement to use of new technologies.

In some cases as in the Ford example in Section 5, faster results can be achieved on existing in-house machines using existing ISV-based software but utilizing multidisciplinary optimization techniques external to the ISV-based software in a parallel computing environment. Several of the problems listed in Section 3 could benefit from use of such strategies. Also there exist several codes available from government labs that could immediately speed up the solution process of some of the submitted problems and then be phased in to the ISV-based software via a cooperative effort between the ISV and the government lab involved.

8 Acknowledgment

I am very grateful to Dr. Jaroslaw Sobieszczanski-Sobieski (Leader- Computational AeroSciences Team, NASA Langley Research Center) for the opportunity to serve as a NASA/ASEE Summer Research Fellow in his group and to investigate large-scale automotive problems. Thanks to James P. Johnson (General Motors) and Dale Shires (Army Research Lab) for suggestions to improve this paper's readability.

References

1. Sobieszczanski-Sobieski, J., Ginsberg, M.: A Preliminary Investigation of Large-Scale High-Performance Computing Applications from the U.S. Automotive Industry, NASA /TM-2001-xxxxxx Computational AeroSciences Team, NASA Langley Research Center, Hampton, VA, to be published, 2001
2. Biedron, R.T., Mehrotra, P., Nelson, M.L., Preston, F.S., Rehder, J.J., Rogers, J.L., Rudy, D.H., Sobieski, J., Storaasli, O.O.: Compute as Fast as the Engineers Can Think! -Ultrafast Computing Team Final Report, NASA/TM-1999-209715, NASA Langley Research Center, Hampton, VA, September 1999
3. Hauser, J. et al.: A Pure Java Parallel Flow Solver. Proceedings, 37th AIAA Aerospace Science Meeting, Paper AIAA 99-0549 (1999)
4. Ginsberg, M.: An Overview of Supercomputing at General Motors Corporation. In: Ames,, K.R., Brenner, A.G. (eds): Frontiers of Supercomputing II: A National Reassessment. volume in the Los Alamos Series in Basic and Applied Sciences, University of California Press, Berkeley (1994) 359-371
5. Ginsberg, M.: Creating an Automotive Industry Benchmark Suite for Assessing the Effectiveness of High-Performance Computers. SAE Trans (J. Passenger Cars). 104, Part 2 (1995) 2048-2057

6. Ginsberg, M.: AUTOBENCH: A 21st Century Vision for an Automotive Computing Benchmark Suite. In: Sheh, M. (ed.): High Performance Computing in Automotive Design, Engineering, and Manufacturing, Cray Research, Inc., Eagan, MN (1997) 67-76
7. Ginsberg, M.: Supercomputer Help Auto Manufacturers Decrease Lead Time. In: Redelfs, A. (ed.): HPC Contributions to Society. Tabor Griffin Communications, San Diego, (November 1998) 74-79
8. Cray Inc.: Ford Installs Two More Cray SV1 Supercomputers, URL: http://www.cray.com/news/0008/ford.html (Aug. 3, 2000)
9. HPCWIRE: MSC LINUX Makes Supercomputing Accessible. Article 19168, HPCWIRE (January 5, 2001)
10 Sobieszczanski-Sobieski, J. et al.: Optimization of Car Body under Constraints of NVH and Crash. AIAA Paper 2000-1521 (2000)
11 Ginsberg, M.: Influences, Challenges, and Strategies for Automotive HPC Benchmarking and Performance Improvement. Par. Comp J. (1999) 1459-1476
12. Ginsberg, M.: Current and Future Status of HPC in the World Automotive Industry. In: Henderson, M.E., Anderson, C.R., Lyons, S.L. (eds.): Object Oriented Methods for Inter-Operable Scientific and Engineering Computing, Society for Industrial and Applied Mathematics (SIAM), Philadelphia, (1999) 1-10
13. HPCWIRE: NEC & ESI Announce Virtual Engine Design System. Article 18918, HPCWIRE (Nov. 17, 2000)
14. HPCWIRE: Computer Simulation Allows "Virtual" Tire. Article 18715, HPCWIRE (Oct. 13, 2000)
15. HPCWIRE: Mazda Selects MTS for Design Improvement Project Article 18775, HPCWIRE (Oct. 20, 2000)
16. HPCWIRE: Ford Teams with UK's Defence Evaluation Agency. Article 19064, HPCWIRE (Dec. 8, 2000)
17. HPCWIRE: Tele-Immersion May Change How We Communicate. Article 18761, HPCWIRE (Oct. 20, 2000)
18. Schulz, M., Ertl,, T., Reuding, T.: Crashing in Cyberspace - Evaluating Structural Behaviour of Car Bodies in a Virtual Environment. Proceedings, IEEE VRAIS'98, (1998) 160-166
19. HPCWIRE: Cray Announces Plans for Alpha Linux Supercluster Systems. Article 19347, HPCWIRE (February 2, 2001)
20. HPCWIRE: The Beowulf Factor in High Performance Computing. Article 19022, HPCWIRE (December 8, 2000)
21. HPCWIRE: Scientific Supercomputing in Germany. Article 18650, HPCWIRE (Oct. 6, 2000
22. HPCWIRE: SC User Group Notes Progress on Plan for Better Perf Tests. Article 60176, HPCWIRE (Nov. 7, 2000)
23. HPCWIRE: SGI and ESI Achieve Unprecedented Computing Power. Article 18871, HPCWIRE (Nov. 3, 2000)
24. HPCWIRE: Cray Inc. and Mcube Team Up on Auto Tests. Article 18839, HPCWIRE (Oct. 27, 2000)
25. HPCWIRE: Cray and MSC Software Team Up. Article 18733, HPCWIRE (Oct. 13, 2000)
26. Mavriplis, D.J.: Parallel Performance Investigations of an Unstructured Mesh Navier-Stokes Solver. ICASE Report No. 2000-13, ICASE, Hampton, VA (March 2000)
27. Foster, I., Kesselman, C. (eds.): The Grid Blueprint for a New Computing Infrastructure. Morgan-Kaufmann, San Francisco, CA (July 1998)

Scalable Large Scale Process Modeling and Simulations in Liquid Composite Molding

Ram Mohan*, Dale Shires, and Andrew Mark

U. S. Army Research Laboratory †
High Performance Computing Division
Aberdeen Proving Ground, MD 21005

Abstract. Various composite structural configurations are increasingly manufactured using liquid composite molding processes. These processes provide unitized structures with repeatability and excellent dimensional tolerances. The new generation of materials and processes, however, pose significant challenges in affordable development and process optimization. Process modeling and simulations, based on physical models, play a significant role in understanding the process behavior and allow for optimizing the process variants in a virtual environment. Large scale composite structures require scalable process modeling and simulation software. An overview of various key issues for such scalable software developments are presented. Several items, including our experiences with the numerical techniques and algorithms for the solution of the representative physical models, are discussed. We also investigate and briefly discuss the various parallel programming models and software development complexities to achieve good performance. In this context, iterative solution techniques for large scale systems, based on a domain decomposition of the problem domain and non-invasive techniques to boost performance of scalable software, are discussed. Preliminary results of the performance of simulations in liquid composite molding are presented. The discussions presented, though in direct reference to scalable process modeling and simulations in liquid composite molding, are directly applicable to any unstructured finite element computations.

1 Introduction

Polymer composite materials have become the material of choice in various new aerospace, marine, and dual-use military and commercial applications. This is due to the high strength-to-weight ratio of these materials. However, manufacture of structural composite materials poses significant problems. These composite materials are made of a fiber matrix and are consolidated together by a polymeric resin. The strength of these composite materials is due to the directional

† This research was made possible by a grant of computer time and resources by the Department of Defense High Performance Computing Modernization Program.

* Guest Researcher

V.N. Alexandrov et al. (Eds.): ICCS 2001, LNCS 2073, pp. 1199–1208, 2001.
© Springer-Verlag Berlin Heidelberg 2001

orientation of the individual fabric layers. Manufacturing techniques should ideally preserve this fiber orientation during processing for each of the repeated composite structural parts. Processes like prepreg and hand layup maintain the high quality of the layer orientations and good bonding between the individual layers. However, these tend to be highly time consuming and labor intensive. New processing methodologies based on liquid molding of composites have evolved over the past decade to provide a production oriented methodology for structural composite configurations. The process allows for repeated production of net-shape composite structures that maintain and preserve the desired fiber orientation in a production environment. One particular type of liquid composite molding is resin transfer molding (RTM) and its related variants. The process involves only a few steps. The first step involves construction of a tool or mold cavity that could be either one- or two-sided. This is followed by setting up a dry fiber preform with multiple individual layers of woven fabrics (normally stitched together) and preforming the dry fabric into a net-shape of the composite structural part configuration. The net-shaped composite structural part configuration is placed inside the mold cavity conforming to the tool surface. The polymeric resin is then injected into the mold cavity where it infiltrates and impregnates the dry fiber preform, thus consolidating into the net-shaped structural composite part configuration after curing and consolidation.

The challenges facing the new generation of material processes for producing these composite structures are how these processes and materials can be cost effectively developed and optimized for various processing conditions. The mold tooling costs in liquid molding applications are high and it is essential to develop optimal processing conditions. Physical process modeling and simulations enable understanding the physical behavior and optimizing the process variables in a virtual environment. The strength of the liquid molding processes such as RTM and its variants is in their ability to manufacture unitized, net-shape, large scale composite configurations. The physical and geometric complexity necessitate a need for scalable simulation software for the process modeling and simulations.

Physical process modeling and simulations begin with the understanding and description of the underlying physical phenomena by means of mathematical model equations representing the various conservation laws, and, appropriate material constitutive relations. The model equations are then discretized by numerical methods such as the finite element method. Computational algorithms play a critical role in obtaining physically and numerically accurate solutions for large scale problems. In particular, in scalable computational analysis, it is critical and important to employ optimal computational methodologies. This is in addition to having optimal data structures, data and processor layout, and communication strategies for scalable high performance computing architectures. A purely finite element based methodology [1,2,3] for modeling and analyzing the pressure driven transient flow through porous fiber media is briefly described.

The development of large scale, parallel scalable software involves a clear trade-off between the amount of work and the information the developer has to perform and provide, and the amount of effort the compiler has to expend to

generate the optimal scalable parallel software code. This depends mainly on parallel programming paradigms. Programming paradigms based on high level parallel languages (High Performance Fortran (HPF), CM-Fortran[4] in CM-5 systems) and parallelizing compilers permit a data parallel mode of computing with no user defined explicit communication across multiple processors. Another parallel programming paradigm involves explicit message passing across the multiple processors. With the development and maturation of the message passing standards from the early 1990s, this approach has gained wider usage for scalable parallel software development. Various issues of the message passing interface (MPI) parallel software developments are discussed and presented. Illustrative examples demonstrating the computational time speed up for multiprocessor runs are presented. Preliminary results comparing the execution times between the HPF and MPI parallel paradigms are provided in the context of these simulations.

2 Process Modeling of Resin Impregnation

The resin impregnation process in liquid molding processes such as RTM involves the flow of a polymeric resin through a fiber network until the network is completely filled. The macroscopic flow of the resin through the complex fiber medium is usually treated as a pressure driven flow through a porous media and is characterized by Darcy's law. Darcy's law relates the macroscopic flow velocity field to the pressure gradient through fluid viscosity and fiber preform permeability as

$$\boldsymbol{u} = \frac{K}{\mu} \bigtriangledown P, \tag{1}$$

where $\boldsymbol{u}$ is the velocity field, μ is the fluid viscosity, and P is the pressure. Permeability K is a measure of the resistance a fluid experiences when flowing through a porous medium and is an important material characteristic in the process flow impregnation simulations. Physically, resin impregnation and flow permeating through a porous media is a free surface moving boundary value problem whereby the field equations and the free surface have to be solved and tracked. In the liquid composite molding processes such as RTM, the primary interest is in the temporal progression of the resin inside a complex mold cavity representing the net-shape composite structural part. Eulerian fixed-mesh approaches are employed for numerical finite element computations of the complex diverging/merging flow front progressions.

2.1 Numerical Methodologies

Computational methodologies employed in process flow modeling simulations for solving the pressure field and the free surface include (1) an explicit finite element-control volume method and (2) the pure finite element method originally developed by Mohan et al. [1,2,3].

In the finite element-control volume technique [5,6,7], the transient resin impregnation problem is treated as a quasi-steady-state problem solving for the quasi-steady incompressible continuity equation. This leads to stringent numerical restrictions based upon the Courant stability conditions. The time step increments are highly restricted across each of the quasi-steady steps ensuring the numerical stability of the quasi-steady approximations based on the Courant stability conditions. Such restrictions increase the number of quasi-steady state steps needed for the analysis. The power of high performance computing lies in its ability to enable practical large scale simulations in a reasonable time. For large scale problem sizes and computational domains, the quasi-steady time increment sizes are extremely small compared to the resolution needed in the application. This makes it impossible to complete large scale simulations, even on scalable high performance computing platforms in any reasonable time. These difficulties are overcome by the pure finite element methodology [1,2,3] briefly described next.

The pure finite element methodology is based on the transient mass conservation law modeling the resin mass balance inside a mold cavity involving the state variable $\Psi(0 \leq \Psi \leq 1)$, where $\Psi = 0$ represents the unimpregnated regions of the dry fiber preform, and $\Psi = 1$ represents the completely impregnated regions of the dry fiber preform. The velocity field is modeled with Darcy's law. The governing transient equation is represented by

$$\frac{\partial \Psi}{\partial t} = \bigtriangledown \cdot \left(\frac{K}{\mu} \bigtriangledown P \right), \tag{2}$$

where μ is the resin viscosity and K is the permeability tensor of the fiber preform. For thin preforms (2.5 D physics) with variations in the in-plane velocity fields, the permeability tensor K is a matrix of order two; for thick preforms, the permeability tensor is a matrix of order three. Based on a Galerkin-weighted residual formulation, and, introducing finite element approximations for both the state variable $\Psi(N_i \Psi_i)$ and the pressure field $P(N_i P_i)$, where N_i is the finite element shape function that depends on the element type employed; P_i, Ψ_i are the nodal values, leads to a discretized system of equations given by

$$C \left[\Psi^{n+1} - \Psi^n \right] + \Delta t \, [K] \, P = \Delta t q. \tag{3}$$

In equation 3, C is the mass matrix representing the pore volume, $[K]$ is the stiffness matrix associated with the pressure field, Δt is the time step size for the transient problem, and q is the force vector representing the injection conditions. The pure finite element methodology solves for the fill factor and the pressure field associated with the finite element nodes in an iterative manner until complete mass conservation is achieved at each time step.

The pure finite element methodology briefly described here does not involve the restrictions of the time step increments seen in the explicit finite element-control volume method for this free surface flow problem and is second-order accurate in time. Rather, it computes the position of the flow front at each of

the discrete time steps that are selected by the analyst. This leads to significant reductions in the computing time required to complete the execution of process modeling and simulations. These reductions are quite dramatic in the case of large-scale computational models for composite structural configurations. This methodology is proven to provide a more physically accurate, computationally faster, and algorithmically better solution strategies for the finite element modeling of the resin impregnation in liquid composite molding processes [1,2,3]. The computational advantage demonstrated by the method in comparison to the finite element-control volume technique for liquid resin impregnation composite molding simulations is solely due to the computational methodology and the algorithmic solution strategy. The computational effectiveness of the strategy is independent of the computational platform. Large-scale process modeling and simulations that were impossible earlier are now possible.

When the thermal and curing effects are considered during the resin impregnation process, the governing model equations based on a volume-averaged energy balance is given by

$$\rho C_p \frac{\partial \hat{T}}{\partial t} + \rho_f C_{pf} \hat{\boldsymbol{u}} \cdot \bigtriangledown \hat{T} = \bigtriangledown \cdot K_T \bigtriangledown \hat{T} + \phi \hat{\dot{G}}. \tag{4}$$

The subscript f denotes the liquid phase, ϕ is the porosity of the fiber medium, and ρ and c_p are the average density and specific heat, respectively. The curing reaction based on species balance can be written as

$$\phi \frac{\partial \hat{\alpha}}{\partial t} + \hat{u} \cdot \bigtriangledown \hat{\alpha} = \phi \hat{R_\alpha}. \tag{5}$$

In most cases, the impregnation phase is completed before the initiation of thermal and cure reactions. The equations are presented here for illustration and the present scalable developments focus only on the resin impregnation phase.

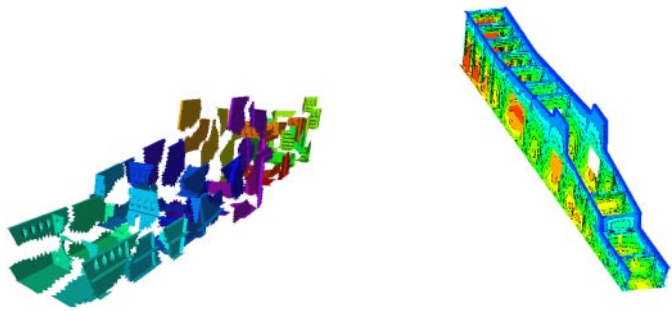
3 Scalable Parallel Software Paradigms

The development of parallel software involves various levels of effort based on the programming paradigms employed. There is a clear correlation between the work that the parallel software developer has to perform and the performance of a parallel code. The paradigms based on parallelizing compilers leave the full responsibility of extracting the parallelism to the compilers and tend to perform poorly, though a minimal level of effort is needed from the developer. At the intermediate level are the parallel languages in the form of HPF, where the developer and the compiler/runtime system share the responsibility of extracting the parallelism. This is based on Fortran directives that allow the developer to express the parallelism (normally data parallel mode in finite element computations) and control the data locality at a very high level. The developer-defined data layout is then utilized in a compiler which generates the low-level details, including the required communications. At the highest level of developer responsibility are explicitly-parallel formulations, such as MPI. Here, the

developer has to explicitly code all the parallelism, synchronization, and masking based on the analysis schemes and interprocedural data requirements. In the MPI based paradigms, parallelism is obtained through a single-program, multiple-data (SPMD) programming model. This is achieved using appropriate decomposition of the computational domain into multiple domains assigned to various processors. Specific interprocessor and required processor masking is specified by the developer with explicit MPI communication and synchronization calls. Both HPF- and MPI- based scalable parallel software have been developed. An illustrative preliminary comparison of total execution times, excluding input and output operations, is presented next.

3.1 HPF and MPI Comparisons

Illustrative analysis resulting from the scalable software developments is shown in figure 1. A domain decomposed partition for MPI-based scalable simulations



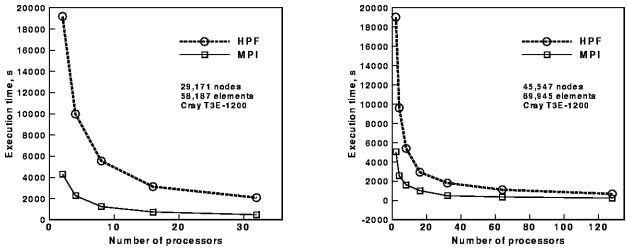
(a) Partitioned Domain for MPI

(b) Temporal resin progression contours

Fig. 1. Scalable process modeling and simulations of a complex structure

is shown in figure 1(a). The nonoverlapping finite element mesh decompositions have been obtained using the graph partitioning software Metis and/or ParMetis [8]. The temporal resin progression contours, based on representative injection conditions for a large, complex aerospace composite structural configuration, are shown in Figure 1(b).

The total execution times for complete analysis based on two finite element mesh configurations are shown in figure 2. The timing comparisons are based on a Cray T3E-1200 system. We chose this system since the HPF compiler employed (Portland Group PGHPF 3.0) was more heavily optimized for this architecture [9]. A preconditioned conjugate gradient iterative solver is employed to solve the linear system of equations. The HPF implementation of the iterative solver is based on a element-by-element formulation where the residuals are determined at the element level before they are assembled to the global level. The MPI implementation is based on a sub-domain assembled (no communication involved within a processor) sparse implementation and is discussed further in the next section. It is clearly seen from figure 2, that the execution time for the HPF



(a) Configuration 1 (b) Configuration 2

Fig. 2. Total execution time comparison of HPF and MPI scalable software

version of scalable software is higher compared to the MPI solver. This illustrates that significantly better performance can be obtained with MPI-based paradigms that require a full developer involvement. However, new approaches to boost performance of data parallel approaches for unstructured mesh problems are coming to fruition. For example, the new PGHPF 3.2 compiler supports asymmetric block data distribution, and the Japan Association for HPF has proposed techniques to reuse communication schedules inside of gather-type operations. Further studies on the computation, communication costs, memory usage, and linear system solver performance are currently in progress.

4 Linear System Solver with MPI

The process modeling and simulations in liquid composite molding involve the solution of a linear system of equations of the form $Ax = b$, based on a sparse, symmetric positive definite matrix. An iterative conjugate gradient algorithm, based on the developments [11] for multiple instruction, multiple data (MIMD) computers, is employed. A similar procedure has been followed and extended to include diagonal preconditioners [10]. The assembled and distributed vector forms are employed. Assembled forms are denoted by $(\hat{\cdot})$ [12]. The iterative procedure is thus

- $\{x_0^{SD}\} = 0$, $\{r_0^{SD}\} = \{b^{SD}\}$
- Do $i = 0, 1, \ldots\ldots$ until convergence

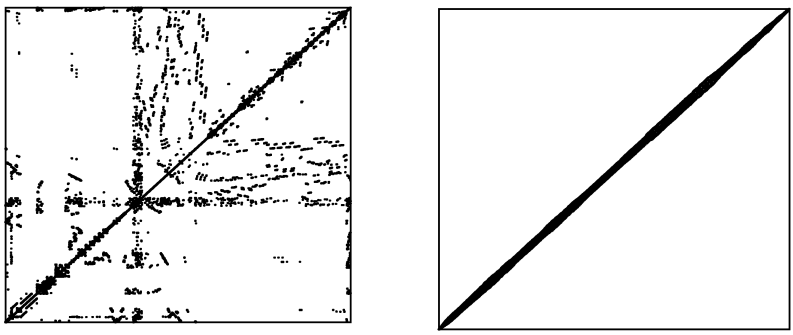
1. If $i = 0$ Go To Step 5
2. $\{u^{SD}\} = \left[A^{SD}\right] \{p^{SD}\}$
   $\beta^{SD} = \{p^{SD}\}^T \{u^{SD}\}$
   $\sigma_{SD} = \beta^{SD}/\gamma$
3. MPI_ALLREDUCE $\frac{1}{\alpha} = \Sigma\sigma^{SD}$
4. $\left\{x^{SD}\right\} = \left\{x^{SD}\right\} + \alpha \left\{p^{SD}\right\}$
   $\left\{r^{SD}\right\} = \left\{r^{SD}\right\} - \alpha \left\{u^{SD}\right\}$
5. SEND $\{r^{SB}\}$; RECEIVE $\{r^{NB}\}$
   $\left\{\hat{r}^{SD}\right\} = \left\{r^{SD}\right\} + \Sigma\{r^{NB}\}$
   $\rho^{SD} = \left\{\hat{r}^{SD}\right\}^T \left\{r^{SD}\right\}$
6. MPI_ALLREDUCE $\gamma_{new} = \Sigma\rho^{SD}$
7. If $(i = 0)$ Go to Step 9
8. If $\left(\frac{\gamma_{new}}{\gamma} < \epsilon\right)$ STOP
9. $\left\{p^{SD}\right\} = \left\{\hat{r}^{SD}\right\} + \frac{\gamma_{new}}{\gamma} \left\{p^{SD}\right\}$
10. $\gamma = \gamma_{new}$

The method is demanding in terms of both communication and computations and is called repeatedly during the analysis runs.

One of the major steps in the previous iterative procedure is matrix-vector multiplication. The sparse matrix is repeatedly multiplied by a vector. The cache performance of SAXPY with the matrix A in sparse format depends on the finite element connectivity. Performance gains are possible by noninvasive techniques

that actually modify the data sets without affecting the physics behind the simulation. The techniques presented next are apropos for reduced instruction set (RISC) based computers currently in use.

Most RISC-based processors utilize various levels (at least two) of cache to overcome the CPU/memory system performance gap [13]. A poorly structured connectivity of an unstructured finite element mesh can lead to poor cache affinity. The sparse matrix A is in a compressed row format, and the node-based computations are based on element connectivity. For instance, matrix-vector multiplication with an element composed of nodes numbered 3, 100, and 200 will require a load of data from the node-based vector *vec* at addresses &*vec*[3], &*vec*[100], and &*vec*[200]. A load of data for *vec*[3] will likely load data into cache for entries *vec*[4] and *vec*[5], but hardly data for nodes 100 and 200. A node renumbering that will improve the proximity of node numbers will also improve the cache efficiency. Various techniques are available to renumber the meshes to promote cache efficiency [14]. Reordering finite element nodes based on the Reverse Cuthill McKee (RCM) scheme [15] significantly improved the closeness of the data for the matrix-vector multiplications. Figure 3(a) shows the distribution of non-zero entries of a matrix A before reordering, and figure 3(b) shows the distribution of entries after reordering. The closeness of the non-



(a) Before reordering (b) After reordering

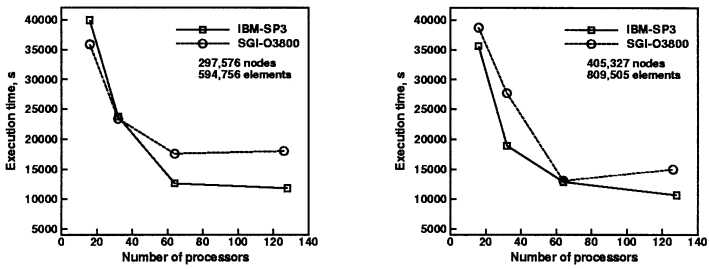
Fig. 3. Distribution of non-zero entries in a finite element sparse matrix A

zero entries after reordering permits loading of all required nodal vectors into the cache. This is important in the case of high performance computing systems with small cache sizes for combined data and instruction. For example, the Cray T3E processors only contain 8KB of primary cache and 96KB of secondary cache. An RCM processed finite element mesh executed roughly 10% faster on the Cray T3E than an unprocessed mesh. However, on machines with a large cache, such as the SGI Origin 3800 with an 8MB secondary cache, the wall clock time is unaffected. A detailed analysis of the underlying memory system, however, did show dramatic improvements in cache performance with reduced cache misses and improved memory performance. It is surmised that the large cache, coupled with optimizations such as software and hardware prefetching, mitigated

wall clock improvements. Further detailed analysis is currently being conducted. Similar approaches improve the data locality in HPF implementations [16].

5 Scalable Simulations

The total execution times based on scalable multiprocessor MPI simulation runs for two different mesh configurations are shown in figure 4. These runs represent different processing conditions for the composite structural configuration shown in figure 1(a). The results imply a good scalability and portability of the scalable



(a) Configuration 1 (b) Configuration 2

Fig. 4. Execution time and speedup from scalable MPI simulations

software, and demonstrate the scalable process modeling and simulation capability to perform realistic simulations in a reasonable time. Large-scale process modeling simulations are now routinely possible for complex composite structural configurations manufactured by liquid molding processes such as RTM and its variants. Further performance studies are currently in progress.

6 Concluding Remarks

Modeling and simulation, coupled with scalable high performance computing, can play a significant role in various new composite manufacturing applications. These techniques, available because of new numerical approaches such as the pure finite element method and large scale parallel computing assets, reduce the risks associated with manufacturing large, geometrically complex composite structural components. The impacts are wide ranging, and applicable to numerous military and commercial uses.

This paper has demonstrated how various parallel methodologies can be applied to complex engineering problems. High level approaches to parallelism, such as data parallelism through HPF, are applicable to this class of problem. These approaches continue to mature through more robust compilers and an enhanced standard. Large scale industrial applications may benefit more from an explicitly parallel approach found in message passing, where the MPI libraries are providing fast solutions to grand-scale problems on numerous parallel architectures.

References

[1] R. V. Mohan, N. D. Ngo, and K. K. Tamma. On a Pure Finite-Element-Based Methodology for Resin Transfer Mold Filling Simulations. *Polymer Engineering and Science*, Vol. 39, no.1, January 1999.

[2] R. V. Mohan, N. D. Ngo, K. K. Tamma, and K. D. Fickie. A Finite Element Based Methodology for Resin Transfer Mold Filling Simulations. In R. W. Lewis and P. Durbetaki, editors, *Numerical Methods for Thermal Problems*, Vol. IX, pp. 1287–1310, Atlanta, GA, July 1995. Pineridge Press.

[3] R. V. Mohan, N. D. Ngo, K. K. Tamma, and D. R. Shires. Three-Dimensional Resin Transfer Molding: Isothermal Process Modeling and Implicit Tracking of Moving Fronts for Thick, Geometrically Complex Composite Manufacturing Applications - Part 2. *Numerical Heat Transfer-Part A, Applications*, Vol. 35, no. 8, 1999.

[4] R. V. Mohan, D. R. Shires, A. Mark, and K. K. Tamma. Advanced Manufacturing of Large Scale Composite Structures : Process Modeling, Manufacturing Simulations and Massively Parallel Computing Platforms. *Journal of Advances in Engineering Software*, Vol. 29, no. (3-6), pp. 249–264, 1998.

[5] C. A. Fracchia, J. Castro, and C. L. Tucker. A Finite Element/Control Volume simulation of Resin Transfer Mold Filling. In *Proc. of the American Society For Composites, 4th technical conference*, pages 157–166, Lancaster, PA, 1989.

[6] M. V. Bruschke and S. G. Advani. A Finite Element/Control Volume Approach to Mold Filling in Anisotropic Porous Media. *Polymer Composites*, Vol. 11, no. 6, pp. 398–405, 1990.

[7] F. Trouchu, R. Gauvin, and D. M. Gao. Numerical Analysis of the Resin Transfer Molding Process by the Finite Element Method. *Advances in Polymer Technology*, 12(4):329–342, 1993.

[8] George Karypis and Vipin Kumar. *METIS: A Software Package for Partitioning Unstructured Graphs, Partitioning Meshes, and Computing Fill-Reducing Orderings of Sparse Matrices.* University of Minnesota and the Army HPC Research Center, 1997.

[9] Portland Group, Major Shared Resource Center Symposium Series, U. S. Army Research Laboratory, 2000.

[10] R. Kanapady. Parallel Implementation of Large Scale Finite Element Computations on a Multiprocessor Machine: Applications to Process Modeling and Manufacturing of Composites. Masters Thesis, 1998. University of Minnesota.

[11] K. H. Law. A Parallel Finite Element Solution Method. *Computers & Structures*, Vol. 23, no. 6, pp. 845–858, 1985.

[12] D. R. Shires, R. V. Mohan, and A. Mark A Study of Parallel Software Development with HPF and MPI for Composite Process Modeling Simulations DOD Users Group Conference, Albuquerque, NM, 2000.

[13] T. Chilimbi, M. Hill, and J. Larus. Making Pointer-Based Data Structures Cache Conscious. *Computer*, vol. 33, no. 12, pp. 67–74, 2000.

[14] G. Kumfert and A. Pothen. Two Improved Algorithms for Envelope and Wavefront Reduction. *BIT*, Vol. 37, no. 3, pp. 559–590, 1997.

[15] E. Cuthill and J. McKee. Reducing the Bandwidth of Sparse Symmetric Matrices. *24th National Conference*. Association for Computing Machinery, pp. 157–172, 1969.

[16] D. R. Shires, R. V. Mohan, and A. Mark. Improving Data Locality and Expanding the Use of HPF in Parallel FEM Implementations. International Conference on Parallel and Distributed Processing Techniques and Applications, LasVegas, NV, 2000.

An Object-Oriented Software Framework for Execution of Real-Time, Parallel Algorithms*

J. Brent Spears and Brett N. Gossage

ERC, Inc.
555 Sparkman Drive, Executive Plaza, Suite 1622 Huntsville, AL 35816
{bspears, bgossage}@rttc.army.mil

Abstract. Infrared (IR) scene projection systems that employ micro-resisters can have significant, non-uniform spatial variation in their output. This non-uniformity causes unwanted artifacts in the projected scene sufficient to prevent accurate sensor testing. To compensate for this non-uniformity, high-speed digital hardware is used to apply non-uniformity correction (NUC) to the images. However, the hardware is very costly and must be custom-built for each projector. With high performance computers, NUC can be implemented in software at a fraction of the hardware cost while still meeting the real-time requirements. The purpose of this paper is to present object-oriented frameworks, implemented in C++, for executing NUC algorithms in parallel in real-time. The frameworks provide abstractions for multi-threading, parallel processing, shared memory, frame-based scheduling, variable interrupt sources, and scheduling disciplines. Results for NUC algorithms on an 8-processor SGI Onyx-2 will also be presented.

1 Introduction

To avoid costly field-testing, infrared (IR) sensors are tested by projecting an infrared scene into its optical aperture and running the system within a real-time simulation. The scene is generated by a micro resister array that converts image pixel values into thermal energy. A difficulty in this conversion process arises from the fact that there can be a significant variation in the electrical resistance values for all of the micro-resistors. This *non-uniformity* causes unwanted artifacts in the projected scene. For the scene projector to be a useable device, a provision for non-uniformity correction (NUC) must implemented. The current solution to the non-uniformity problem is to selectively adjust the intensity value of the input image on a resistor by resistor basis. This must be done for each pixel in the image at video frame rates. The present implementation of NUC relies on costly high-speed digital hardware. The goal of this effort was to develop a new class of scalable and reusable NUC algorithms which will perform real-time

* Funding for this project was provided by the Common High-Performance Software Support Initiative (CHSSI) in cooperation with the Redstone Technical Test Center (RTTC) Huntsville, AL.

V.N. Alexandrov et al. (Eds.): ICCS 2001, LNCS 2073, pp. 1209–1218, 2001.
© Springer-Verlag Berlin Heidelberg 2001

NUC for scene projectors using generic, parallel computing resources. This paper describes object-oriented frameworks for applying NUC algorithms in real-time. We first present the general design of the system. Next, we will describe the libraries and classes used in the design, particularly the two main components - the real-time frame scheduler and the NUC algorithms. Finally, we present results from testing the algorithms on a multi-processor computer.

2 General Design

There are three main processes in the system - a scene generator, NUC applicator, and a display process. All three processes operate in parallel using a ring buffer with three images (see Figure 1). This creates a pipeline in which the scene generator provides an image, the NUC process then applies the algorithms to the image, and the display process takes the resulting image and displays it to the screen. The NUC process is further divided into threads that are responsible for a slice of the image. Each thread is run exclusively and without interruption on separate processors. A drawback to this design is a two-frame latency (typical for any pipeline). All three processes are synchronized to a particular frame rate (in this case, the vertical refresh rate).

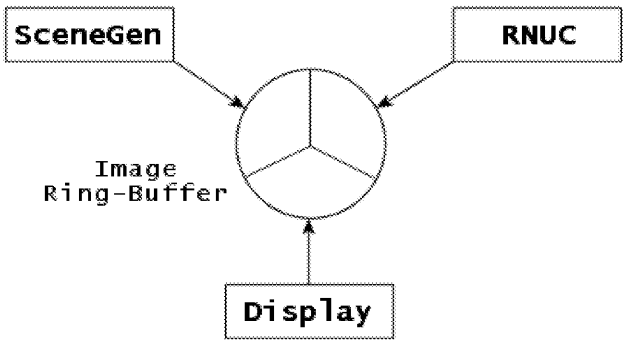


Fig. 1. Processes and Ring-buffer

3 Libraries

The real-time NUC (RNUC) software is based on a core set of reusable libraries. The C++ utility library and the inter-process communication and control (IPCC) library are the core libraries reused by the real-time software system.

3.1 C++ Utility Library

The C++ utility library contains a variety of utility classes such as arrays and strings. One of the more important classes used in the real-time system is the ring buffer adapter. This class is an standard template library (STL[1]) adapter which provides circular iterators (i.e., a ring). In the real-time software system, the ring buffer allows multiple images to be operated on in parallel. Other important features used by the system are the `MemoryPool` class and the `pool_alloc` template class. The `MemoryPool` class serves as an interface for a pool of memory (could be shared or not). It contains member functions such as `allocate` and `deallocate`. The `pool_alloc` template class is the allocator for a `MemoryPool`. The allocator was designed and modeled after the STL allocator in order to make it compatible with the STL. This allows STL constructs (such as lists, queues, and vectors) to be used in a `MemoryPool` (possibly shared memory) by instantiating them with this allocator. The allocator can also be used with other classes that use an STL-compatible allocator.

3.2 Inter-process Communication and Control (IPCC) Framework

The IPCC is a cross-platform library that provides multi-threading and inter-process communication support. The platform independence is achieved by using the *bridge pattern*[GE1][2]. The bridge pattern separates the interface from the implementation. This allows the programmer to code using the interface without worrying about the implementation. This pattern not only supports platform independence, but also allows different types of implementation for the same platform. Several basic inter-process communication constructs are used in the real-time software system. These include threads, mutexes, semaphores, and condition variables. The key capability used by the real-time system is the shared arena.

Shared Arena The `SharedArena` class provides platform-independent support for inter-process communication via shared memory. Platform-specific shared memory capabilities are provided by implementation classes as part of a *bridge pattern* [GE1]. A `SharedArena` supports dynamic memory allocation/deallocation and a single, process-shared user data pointer. The interface semantics of a `SharedArena` are similar to the memory-mapped file interfaces provided by POSIX. Since `SharedArena` is a subclass of `MemoryPool`, it can be combined with the `pool_alloc` template class (effectively creating a shared memory allocator) to support shared C++ objects and collections.

[1] The STL has been incorporated into the C++ Standard Library, but we will use "STL" for the sake of brevity.

[2] A variation of the pattern was used in cases where the constructs are allocated in shared memory.

4 Real-Time Frame Scheduler (Rt) Framework

Rt is a portable real-time frame scheduler library. It defines an interface that permits execution of threads on multiple processors at a specified frame rate. The *bridge pattern*[GE1] is used to separate the interface from the implementation. Therefore, the behavior of Rt can be modified with a new implementation. For example, we have developed a simulated real-time implementation based on the *pthreads* library. This implementation is useful when testing algorithms for correctness during development. The class diagram for Rt is shown in Figure 2.

4.1 Frame Schedulers

The Rt framework interface is based heavily on SGI's REACT/Pro [GE1] interface (our first implementation layer was based on REACT/Pro). A frame scheduler is created and assigned to control a unique processor. The scheduler will exclusively control the processor by restricting and isolating the processor so that only threads assigned to that scheduler will be permitted to execute on that processor. There is always one master frame scheduler and an optional number of slave frame schedulers (slaves are needed when using more than one CPU).

4.2 Interrupt Source

The master frame scheduler manages the frame rate for itself and any slave frame schedulers. The frame rate is defined by an interrupt source. Interrupt source availability is platform dependent. Rt supports several different types of interrupt sources, but all may not be valid on a given platform (availability may be determined at runtime with function calls). An example of an interrupt source is a real-time clock timer or an interrupt from some hardware device. The interrupt source defines the length of a minor frame. One or more minor frames define a major frame (the frame rate). During each minor frame, some work is allowed to be performed by the scheduler.

4.3 Threads

Threads accomplish the work to be done during a minor frame. The programmer simply derives his thread class from `RtThread` and defines his work in the `RtThread::work()` member function. All interaction between a thread and a frame scheduler occurs in the `RtThread::process()` listed below.

```
void RtThread::process()
{
// Wait for frame scheduler to enqueue the thread...
   m_enqueuedSemaphore.wait();

// Join the frame scheduler...
   m_scheduler->joinScheduler();
```

```
// Main loop...
   while( !m_finished )
   {
   // Work function hook...
      this->work();

   // Yield the cpu and suspend...
      m_scheduler->yield();

   } // end while

} // end process
```

Threads are enqueued to minor frames of a scheduler and are executed during that particular minor frame. Multiple threads may be enqueued to the same minor frame and they are executed in the ordered enqueued. A thread must complete before the end of the minor frame it was enqueued to. If not, an **overrun** is said to have occurred. And if a thread is never started[3] during its minor frame, an **underrun** is said to have occurred.

4.4 Scheduling Disciplines

By default, Rt will report overruns and underruns by calling an error handler installed by the programmer (or a default one which does nothing). The programmer can control whether overruns and underruns are reported on a per-thread basis using the scheduling disciplines described below.

Strict real-time Thread must be started and completed during a minor frame.
Overrunnable Thread is allowed to overrun.
Underrunnable Thread is allowed to underrun.
Continuable Allows a thread to be started just once and execute throughout a major frame.
Background Thread is started only after all other threads have completed.

4.5 Creating Schedulers

The basic steps for creating a frame scheduler are as follows:

- create slave schedulers
- create master frame scheduler
- enqueue worker threads to minor frames
- start master scheduler
- wait for worker threads to complete

[3] A thread can underrun if the threads enqueued before this one use the entire minor frame

The following code demonstrates the steps for creating and using the Rt frame schedulers:

```
// Create slaves...
    RtSlaveFrameScheduler slave1( 2 );   // cpu 2
    RtSlaveFrameScheduler slave2( 3 );   // cpu 3

// Add to slave set...
    RtSlaveSchedSet slaveSet;
    slaveSet.add( &slave1 );
    slaveSet.add( &slave2 );

// Create interrupt source (50 milliseconds)...
    RtIntrSource intrSource( rt_intr_cctimer, 50000 );

// Create master scheduler on cpu 1 with 1 minor frame...
    RtMasterFrameScheduler master( 1, intrSource, 1, slaveSet );

// Create worker threads and add to thread set (for
   convenience)...
    ThreadSet threads;
    TestThread worker_thread1;
    TestThread worker_thread2;
    TestThread background_thread;
    threads.add( &worker_thread1 );
    threads.add( &worker_thread2 );

// Enqueue worker threads to slave schedulers...
    RtSchedDiscipline disc( (rt_disc_t)( rt_disc_realtime ) );
    slave1.enqueue( &worker_thread1, 0, disc );
    slave2.enqueue( &worker_thread2, 0, disc );

// Enqueue background thread to master scheduler...
    disc.set( (rt_disc_t)( rt_disc_background ) );
    master.enqueue( &background_thread, 0, disc );

// Start the master scheduler (this also starts slaves)...
    master.start();

// Wait for worker threads to finish (using the thread set)...
    threads.join();
```

4.6 REACT/Pro Implementation

As previously stated, Rt is based on SGI's REACT/Pro [CD1]. This implementation is a hard real-time implementation for SGI multi-processor machines

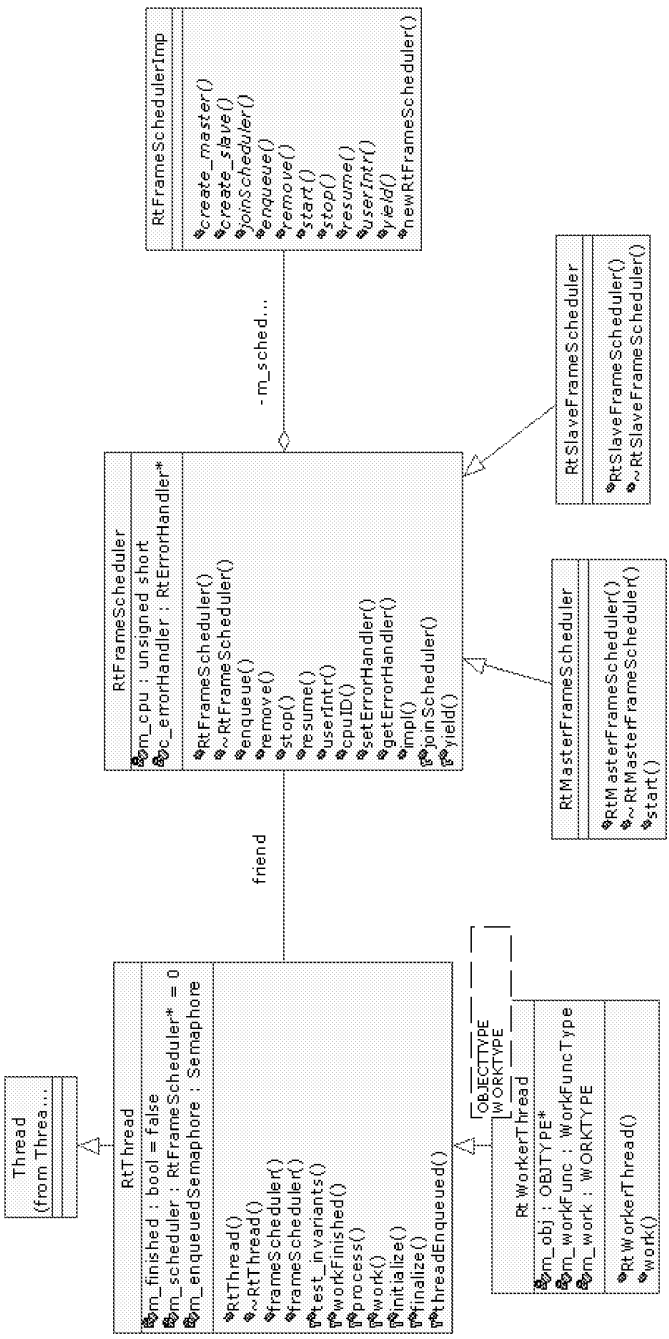


Fig. 2. Rt Class Diagram

with REACT/Pro installed. The interrupt sources available are the CPU-specific timer, real-time clock, vertical retrace interrupt, hardware interrupts, and a user-defined interrupt (recommended for debugging purposes only). This implementation requires certain privileges and kernel reconfiguration.

4.7 Simulated Real-Time Implementation

To test the correct execution of algorithms absent the need for real-time execution, it is only necessary to provide for the scheduling of threads. This capability is provided by implementing a *simulated real-time* frame scheduler. This is accomplished by creating a simulated frame scheduler implementation class within the bridge pattern structure already in place. The simulated real-time implementation allows testing of algorithms on multi-user and single user machines without the need for super user privileges and reconfigured kernels.

5 Non-uniformity Correction (NUC) Algorithms

The input response of each pixel over the resister array must be measured over a number of input values and its response function estimated. Lookup tables (LUT's), least squares fits to polynomials, or non-linear functions are potential models for the response function. In each case, a set of coefficients is given for each pixel over the entire array. While each pixel may have it's own response function, at present it is considered to be independent of its neighboring pixel's input. This means that image non-uniformity correction is a pixel-by-pixel transformation on an image.

5.1 Image Partitioning

Since the NUC process is pixel-by-pixel, the application of the correction is highly data parallel and the image may be divided into sub-images for processing by independent threads. But, that image partitioning cannot be completely arbitrary if false cache sharing and memory paging are to be avoided. Since C/C++ arrays are row-dominant, the image is divided into row strips so that each sub-image is composed of pixels that are contiguous in memory. In addition, the Rt framework restricts the thread CPU to only that thread and no load balancing schemes such as a work pile are necessary. The number of image strips is the same as the number of threads.

5.2 NUC Function Objects

The software must be able to apply arbitrary correction functions with minimal impact on the software. Due to conflicting requirements for both speed and accuracy, a wide variety of pixel mapping functions may be applied. To adapt to different NUC functions with minimal changes to the RNUC code, the RNUC

software employs *function objects* (a.k.a. *functors*) [BS1]. A function object maintains internal state(such as the NUC function coefficients) while providing the semantics of a function call. To separate the algorithms from the type of image data processed, the function objects are implemented as C++ templates. This represents an application of *generic programming* in combination with the object oriented techniques presented so far. The class template `NucOperator` is provided as a common base class for NUC algorithm classes. It has a single template argument that is the image type used by the imaging system. This allows easy adaptation to 8-bit or 16-bit imagery. `NucOperator` defines the pure virtual functions `apply` and `loadCoeffecients` as hooks for derived NUC operator classes to define their own correction algorithm and coefficient input.

```
template< class ImageType >
class NucOperator
{
   public:
      typedef typename ImageType::value_type PixelType;
      typedef IP_ROI RoiType;

   /** Apply the action of this operator over a given roi. */
      virtual void apply( const ImageType& input, ImageType& output,
                          const RoiType& roi ) = 0;

   /** Define operator () to allow use as a functor. */
      void operator () ( const ImageType& input, ImageType& output,
                         const RoiType& roi )
                       { this->apply(input,output,roi); }

   /** Load the correction coefficients.
   @param filename the name of the input file. */
      virtual void loadCoefficients( const cstring& filename ) = 0;

};// end class NucOperator
```

A Look Up Table (LUT) image corrector can now be defined by subclassing from `NucOperator`.

```
template< class ImageType >
class LutCorrector : public NucOperator<ImageType>
{
   public:
      typedef NucOperator<ImageType> Base;

   /** Override NucOperator::apply. */
      void apply( const ImageType& input, ImageType& output,
                  const RoiType& roi );

   /** Override NucOperator::loadCoefficients. */
```

```
    void loadCoefficients( const cstring& filename );

};// end class LutCorrector
```

5.3 Results

We now present results from our tests running on an 8-processor SGI Onyx-2. The NUC algorithms were applied to two images (16-bit pixel values) using up to 5 processors (see Table 1).

Table 1. RNUC Test Results

Image	Cpu's	Frame Rate (Hz)
256x256x1	1	78
	2	150
	3	222
	4	282
	5	340
512x512x1	1	20
	2	38
	3	60
	4	75
	5	90

6 Conclusion

We have demonstrated that, for our RNUC application, generic high-performance systems can remove the need for expensive, custom-built hardware. It is an often-repeated myth that making software object-oriented necessarily makes it slower. This is simply not the case and we have shown that high-performance, real-time requirements can be met using a combination of object-oriented and generic-programming techniques. In fact, it is their synergistic combination that can provide the best solution to the need for both flexible and efficient software.

References

[GE1] Gamma, Erich, et al.: Design Patterns: Elements of Reusable Object-Oriented Software (1995)

[CD1] Cortesi, David and Thomas, Susan: REACT Real-Time Programmer's Guide (1998)

[BS1] Stroustrup, Bjarne: The C++ Programming Language (2000)

A Multiagent Architecture Addresses the Complexity of Industry Process Re-engineering

John Debenham
University of Technology, Sydney
debenham@it.uts.edu.au

Abstract. Industry processes are the trans-corporate business processes required to support the e-business environment. Industry process re-engineering is the re-engineering of trans-corporate business processes as electronically managed processes. Industry process re-engineering is business process re-engineering on a massively distributed scale. Industry processes will not be restricted to routine workflows that follow a more-or-less fixed path; they will include complex processes for which their future path may be unknown at each stage in their existence. So a management system for industry processes should be both highly scalable and should be able to deal with such complex processes. A multiagent process management system is described that can manage processes of high complexity. This system is built from interacting autonomous components so achieving system scalability.

1. Introduction

Business to Business (B2B) e-commerce is driving a new generation of Internet applications that can dramatically automate trans-corporate industry processes only if the business systems and data that drive these industry processes are integrated across the component organisations [1]. Here *industry processes* includes both business transactions, and business processes and workflows. These Internet applications can only automate industry processes if there is a method to describe collaborative processes across organisations and to provide data interoperability. Improvements in process management can only be achieved through automation. Automation of processes leads to faster cycle times, reduced overhead and more competitive offerings. *Industry process re-engineering* is the re-engineering of trans-corporate processes as electronically managed processes. Companies that have implemented this e-business vision are saving tens of millions of dollars per year [2]. Industry process re-engineering must address the four issues of complexity, interoperability, communication and management:

- *complexity* of industry processes refers to their nature which includes *all* trans-corporate processes from routine workflows to high-level emergent processes [3];
- *interoperability* is an issue due to the heterogeneity of the diverse systems across a trading community. These systems vary in the applications that manage them and in the data formats that they employ;

V.N. Alexandrov et al. (Eds.): ICCS 2001, LNCS 2073, pp. 1219–1228, 2001.
© Springer-Verlag Berlin Heidelberg 2001

- the *communication* and messaging infrastructure chosen will operate in a mission-critical environment, and
- process *management* which is responsible for tracking the automated trans-corporate processes that may include processes unique to individual trading partners and will probably involve a wide range of process steps that must "make sense" to all involved.

That is, industry process re-engineering must deliver a secure, scalable and reliable solution for running a company's most critical core business processes. The complex nature of industry processes is considered here.

High-level emergent processes are business processes that are not predefined and are ad hoc. These processes typically take place at the higher levels of organisations [1], and are distinct from production workflows [2]. Emergent processes are opportunistic in nature whereas production workflows are routine. How an emergent process will terminate may not be known until the process is well advanced. Further, the tasks involved in an emergent process are typically not predefined and *emerge* as the process develops. Those tasks may be carried out by collaborative groups as well as by individuals [4]. For example, in a manufacturing organisation an emergent process could be triggered by "lets consider introducing a new product line for the US market".

From a process management perspective, emergent processes contain "knowledge-driven" sub-processes and conventional "goal-driven" sub-processes [5]. The management of a *knowledge-driven process* is guided by its 'process knowledge' and 'performance knowledge' and *not* by its goal which may not be fixed and may mutate. On the other hand, the management of a *goal-driven process* is guided by its goal which is fixed, although the individual corporations involved in an industry process may not achieve such a fixed goal in the same way.

Multiagent technology is an attractive basis for industry process re-engineering [6]. A multiagent system consists of autonomous components that interact with messages. The scalability issue is "solved"—in theory—by establishing a common understanding for inter-agent communication and interaction. Specifying an inter-agent communication protocol may be tedious but is not technically complex. Standard XML-based ontologies will enable data to be communicated freely [1] but much work has yet to be done on standards for communicating expertise. Specifying the agent interaction protocol is a more complex as it in effect specifies the common understanding on the basis of which the whole system will operate. A multiagent system to manage "goal-driven" processes is described in [7]. In that system each human user is assisted by an agent which is based on a generic three-layer, BDI hybrid agent architecture. The term *individual* refers to a user/agent pair. That system has been extended to support knowledge-driven processes and so to support emergent process management and the full range of industry processes. The general business of managing knowledge-driven sub-processes is illustrated in Fig. 1, and will be discussed in Sec. 2. Any process management system should address the "process knowledge" and the "performance knowledge. *Process knowledge* is the wisdom that has been accumulated,

particularly that which is relevant to the process instance at hand. *Performance knowledge* is knowledge of how effective people, methods and plans are at achieving various things. Sec. 3 discusses the management of the process knowledge. Sec. 4 describes the performance knowledge which is communicated between agents in contract net bids for work. Sec. 5 describes the agent interaction mechanism.

2. Industry processes

Following [8] a *business process* is "a set of one or more linked procedures or activities which collectively realise a business objective or policy goal, normally within the context of an organisational structure defining functional roles and relationships". Implicit in this definition is the idea that a process may be repeatedly decomposed into linked sub-processes until those sub-processes are "activities" which are atomic pieces of work. [viz (op.cit) "An *activity* is a description of a piece of work that forms one logical step within a process."]. A particular process is called a (process) *instance*. An instance may require that certain things should be done; such things are called *tasks*. A *trigger* is an event that leads to the creation of an instance. The *goal* of an instance is a state that the instance is trying to achieve. The *termination condition* of an instance is a condition which if satisfied during the life of an instance causes that instance to be destroyed whether its goal has been achieved or not. The *patron* of an instance is the individual who is responsible for managing the life of that instance [9]. At any time in a process instance's life, the *history* of that instance is the sequence of prior sub-goals and the prior sequence of knowledge inputs to the instance. The history is "knowledge of all that has happened already".

From a process management viewpoint, industry processes can be seen as consisting of sub-processes that are of one of the three following types:

- A *task-driven process* has a unique decomposition into a—possibly conditional—sequence of activities. Each of these activities has a goal and is associated with a task that "always" achieves this goal. Production workflows are typically task-driven processes.
- A *goal-driven process* has a process goal, and achievement of that goal is the termination condition for the process. The process goal may have various decompositions into sequences of sub-goals where these sub-goals are associated with (atomic) activities and so with tasks. Some of these sequences of tasks may work better than others, and there may be no way of knowing which is which [3]. A task for an activity may fail outright, or may be otherwise ineffective at achieving its goal. In other words, process failure is a feature of goal-driven processes. If a task fails then another way to achieve the process goal may be sought.
- A *knowledge-driven process* has a process goal, but the goal may be vague and may mutate [10]. Mutations are determined by the process patron, often in the light of knowledge generated during the process. After performing a task in a knowledge-

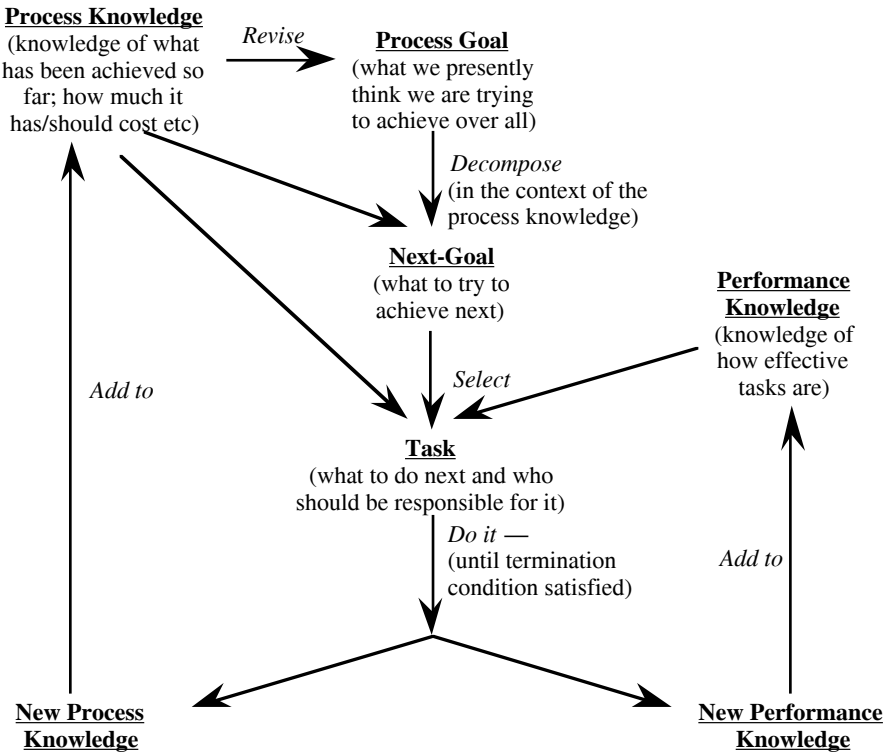


Fig. 1. Knowledge-driven process management (a simplified view)

driven process, the "next goal" is chosen by the process patron. This choice is made using general knowledge concerning the process—called the *process knowledge*. The process patron then chooses the tasks to achieve that next goal. This choice may be made using general knowledge about the effectiveness of tasks—called the *performance knowledge*. So in so far as the process goal gives direction to goal-driven—and task-driven—processes, the growing body of process knowledge gives direction to knowledge-driven processes. The management of knowledge-driven processes is considerably more complex than the other two classes of process, see Fig. 1. But, knowledge-driven processes are "not all bad"—they typically have goal-driven sub-processes.

Task-driven processes may be managed by a simple reactive agent architecture based on event-condition-action rules [5]. Goal-driven processes may be modelled as state and activity charts [11] and managed by plans that can accommodates failure [12]. Such a planning system may provide the deliberative reasoning mechanism in a BDI agent

architecture [12] and is used in a goal-driven process management system [7] where tasks are represented as plans for goal-driven processes. But the success of execution of a plan for a goal-driven process is not necessarily related to the achievement of its goal. One reason for this is that an instance may make progress outside the process management system—two players could go for lunch for example. So each plan for a goal-driven process should terminate with a check of whether its goal has been achieved.

Managing knowledge-driven processes is rather more difficult, see Fig. 1. The role of the process knowledge is described in Sec. 3 and the role of the performance knowledge is described in Sec. 4.

3. Process knowledge and goals

Process knowledge is the wisdom that has been accumulated, particularly that which is relevant to the process instance at hand. For knowledge-driven processes the management of the process knowledge is shown on the left-hand side of Fig. 1. For knowledge-driven processes, management of the process knowledge is impractical.

The process knowledge in any real application includes an enormous amount of general and common sense knowledge. For example, the process trigger "the time is right to look at the US market" may be based on a large quantity of empirical knowledge and a fund of experiential knowledge. So the system does not attempt to represent the process knowledge in any way; it is seen to be largely in the heads of the users. The system does assist in the maintenance of the process knowledge by ensuring that any virtual documents generated during an activity in a knowledge-driven sub-process are passed to the process patron when the activity is complete. Virtual documents are either interactive web documents or workspaces in the LiveNet workspace system which is used to handle virtual meetings and discussions.

The system records, but does not attempt to understand the process goal. Any possible revisions the process goal are carried out by the patron without assistance from the system. Likewise the decomposition of the process goal to decide "what to do next"—the next-goal. It may appear that the system does not do very much at all! If the next-goal is the goal of a goal-driven process—which it may well be—then the system may be left to manage it as long as it has plans in its plan library to achieve that next-goal. If the system does not have plans to achieve such a goal then the user may be able to quickly assemble such a plan from existing components in the plan library. The organisation of the plan library is a free-form, hierarchic filing system designed completely by each user. Such a plan only specifies what has to be done at the host agent. If a plan sends something to another agent with a sub-goal attached it is up to that other agent to design a plan to deal with that sub-goal. If the next-goal is the goal of a knowledge-driven process then the procedure illustrated in Fig. 1 commences at the level of that goal.

So for this part of the procedure, the agent provides assistance with updating the process knowledge, and if a next-goal is the goal of a goal-driven sub-process then the system will manage that sub-process, perhaps after being given a plan to do so.

4. Performance knowledge

Performance knowledge is knowledge of how effective people, methods and plans are at achieving various things. For knowledge-driven processes the management of the performance knowledge is shown on the left-hand side of Fig. 1. Performance knowledge is substantially ignored by many workflow management systems. It is crucial to the efficient management of industry processes. The performance knowledge is used to support task selection—ie who does what—through inter-agent negotiation, see Sec. 5. So its role is a comparative one; it is not required to have absolute currency. With this use in mind, the *performance knowledge* comprises performance statistics on the operation of the system down to a fine grain of detail. These performance statistics are proffered by an agent in bids for work. To evaluate a bid, the receiving agent evaluates its meaning of payoff in terms of these statistics. If a parameter, p, can reasonably be assumed to be normally distributed, the estimate for the mean of p, μ_p, is revised on the basis of the i'th observation ob_i to $\mu_{p_{new}} = (1-\alpha) \times ob_i + \alpha \times \mu_{p_{old}}$ which, given a starting value $\mu_{p_{initial}}$, and some constant α, $0 < \alpha < 1$, approximates the geometric mean $\frac{\sum_{i=1}^{n} \alpha^{i-1} \times ob_i}{\sum_{i=1}^{n} \alpha^{i-1}}$ where i = 1 is the most recent observation. In the same way, an estimate for $\sqrt{\frac{2}{\pi}}$ times the standard deviation of p, σ_p, is revised on the basis of the i'th observation ob_i to $\sigma_{p_{new}} = (1-\alpha) \times | ob_i - \mu_{p_{old}} | + \alpha \times \sigma_{p_{old}}$ which, given a starting value $\sigma_{p_{initial}}$, and some constant α, $0 < \alpha < 1$, approximates the geometric mean $\frac{\sum_{i=1}^{n} \alpha^{i-1} \times | ob_i - \mu_p |}{\sum_{i=1}^{n} \alpha^{i-1}}$. The constant α is chosen on the basis of the stability of the observations. For example, if $\alpha = 0.85$ then "everything more than twenty trials ago" contributes less than 5% to the weighted mean; if $\alpha = 0.70$ then "everything more than ten trials ago" contributes less than 5% to the weighted mean, and if $\alpha = 0.50$ then "everything more than five trials ago" contributes less than 5% to the weighted mean.

Each individual agent/user pair maintains estimates for the three parameters: *time*, *cost* and *likelihood of success* for the execution of all of its plans, sub-plans and activities. "All things being equal" these three parameters are assumed to be normally

distributed—the case when "all things are *not* equal" is considered below. *Time* is the total time taken to termination. *Cost* is the actual cost of the of resources allocated. The *likelihood of success* observations are binary—ie "success" or "fail"—so this parameter is binomially distributed, and is approximately normally distributed under the standard conditions.

Unfortunately, *value* is often very difficult to measure. For example in assessing the value of an appraisal for a bank loan, if the loan is granted then when it has matured its value may be measured, but if the loan is not granted then no conclusion may be drawn. The value of sub-processes are typically "less measurable" than this bank loan example. Although some progressive organisations employ experienced staff specifically to assess the value of the work of others. The existing system does not attempt to measure *value*; each individual represents the perceived *value* of each other individual's work as a constant for that individual.

Finally, the *allocate* parameter for each individual is the amount of work w_i^j, allocated to individual j in discrete time period i. In a similar way to *time* and *cost*, the mean *allocate* estimate for individual j is made using $\text{allocate}_{\text{new}} = (1 - \alpha) \times w^j + \alpha \times \text{allocate}_{\text{old}}$, where w^j is the most recent observation for individual j. In this formula the weighting factor α is chosen on the basis of the number of individuals in the system, and the relationships between the length of the discrete time interval and the expected length of time to deal with the work. The allocate parameter does not represent workload. It is not normally distributed and its standard deviation is not estimated. An estimate of *workload* is given by (allocations in) – (allocations out). The *allocate* and *value* estimates are associated with individuals. The *time*, *cost* and *likelihood of success* estimates are attached to plans.

The three parameters *time*, *cost* and *likelihood of success* are assumed to be normally distributed subject to "all things being equal". One virtue of the assumption of normality is that it provides a basis on which to query unexpected observations. Having made observation ob_{i+1} for parameter p, estimates for μ_p and σ_p are calculated. Then the next observation, ob_i, should lie in the confidence interval: $(\mu_p \pm \alpha \times \sigma_p)$ to some chosen degree of certainty. For example, this degree of certainty is 95% if $\alpha = 1.645$. The set of observations $\{ob_i\}$ can progressively change without individual observations lying outside this confidence interval; for example, an individual may be gradually getting better at doing things. But if an observation lies outside this confidence interval then there is grounds, to the chosen degree of certainty, to ask why it is outside.

Inferred explanations of *why* an observation is outside expected limits may sometimes be extracted from observing the interactions with the users and other agents involved. For example, if Person X is unexpectedly slow in attending to a certain process instance then a simple interchange with X's agent may reveal that Person X will be working on the company's annual report for the next six days; this may be one reason for the unexpected observation. Inferred knowledge such as this gives *one*

possible cause for the observed behaviour; so such knowledge enables us to *refine*, but *not* to *replace*, the historical estimates of parameters.

The measurement ob_i may lie outside the confidence interval for four types of reason:

1) there has been a permanent change in the environment or in the process management system—the measurement ob_i is now the expected value for μ_p—in which case the estimates $\mu_{p_{old}}$ and $\sigma_{p_{old}}$ should be re-initialised.
2) there has been a temporary change in the environment or in the process management system and the measurements $\{ob_i\}$ are expected to be perturbed in some way for some time—in which case the reason, Γ, for this expected perturbation should be sought. For example, a new member of staff may have been delegated the responsibility—temporarily—for this sub-process. Or, for example, a database component of the system may be behaving erratically.
3) there has been no change in the environment or in the process management system and the unexpected measurement ob_i is due to some feature γ that distinguishes the nature of this sub-process instance from those instances that were used to calculate $\mu_{p_{old}}$ and $\sigma_{p_{old}}$. In other words, what was thought to be a single sub-process type is really two or more different—but possibly related—process types. In which case a new process is created and the estimates $\mu_{p_{old}}$ and $\sigma_{p_{old}}$ are initialised for that process.
4) there has been no change in the environment or in the process management system and the nature of the most recent process instance is no different from previous instances—the unexpected measurement ob_i is due to—possibly combined—fluctuations in the performance of individuals or other systems.

In option 2) above the reason Γ is sometimes inferred by the system itself. This has been achieved in cases when a user appears to be preoccupied working on another task. If the reason Γ is to be taken into account then some forecast of the future effect of Γ is required. If such a forecast effect can be quantified—perhaps by simply asking a user—then the perturbed values of $\{ob_i\}$ are corrected to $\{ob_i \mid \Gamma\}$ otherwise the perturbed values are ignored.

5. Agent Interaction

This section concerns the selection of a task for a given now-goal as shown in the middle of Fig. 1. The selection of a plan to achieve a next goal typically involves deciding *what* to do and selecting *who* to ask to assist in doing it. The selection of what to do and who to do it can not be subdivided because one person may be good and one form of task and bad at others. So the "what" and the "who" are considered together. The system provides assistance in making this decision. Sec. 4 describes how performance knowledge is attached to each plan and sub-plan. For plans that involve one individual only this is done for instantiated plans. That is there are

estimates for each individual and plan pair. In this way the system offers advice on choosing between individual A doing X and individual B doing Y. For plans that involve more than one individual this is done by explicitly delegating the responsibility for populating that plan. So if a plan involves forming a committee then it is embedded in a plan that gives an individual the responsibility for forming that committee, and then estimates are gathered for the performance of the second of these.

There are two basic modes in which the selection of "who" to ask is done. First the *authoritarian* mode in which an individual is told to do something. Second the *negotiation* mode in which individuals are asked to express an interest in doing something. This second mode is implemented using contract nets with focussed addressing [13] with inter-agent communication being performed in KQML [14]. When contact net bids are received the successful bidder has to be identified. So no matter which mode is used, a decision has to be made as to whom to select. The use of a multiagent system to manage processes expands the range of feasible strategies for delegation from the authoritarian strategies described above to strategies based on negotiation between individuals. Negotiation-based strategies that involves negotiation for each process instance are not feasible in manual systems for every day tasks due to the cost of negotiation. If the agents in a multiagent system are responsible for this negotiation then the cost of negotiation is may be negligible.

If the agent making a bid to perform a task has a plan for achieving that task then its user may permit the agent to construct a bid automatically. As the bids consist of six meaningful quantities, the user may opt to construct a bid manually. A bid consists of the five pairs of real numbers (Constraint, Allocate, Success, Cost, Time). The pair *constraint* is an estimate of the earliest time that the individual could address the task—ie ignoring other non-urgent things to be done, and an estimate of the time that the individual would normally address the task if it "took its place in the in-tray". The pair Allocate is the mean of allocations-in and the mean of allocations-out. The pairs Success, Cost and Time are estimates of the mean and standard deviation of the corresponding parameters as described above. The receiving agent then:

- attaches a subjective view of the *value* of the bidding individual;
- assesses the extent to which a bid should be downgraded—or not considered at all—because it violates process constraints, and
- selects an acceptable bid, if any, possibly by applying its 'delegation strategy'.

If there are no acceptable bids then the receiving agent "thinks again".

6. Conclusion

Managing trans-corporate industry processes involves managing processes of three distinct types [5]. The management of knowledge-driven processes is not widely understood and has been described here. A multi-agent system manages goal-driven processes and supports the management of knowledge-driven processes [7]. The

conceptual agent architecture is a three-layer BDI, hybrid architecture. During a process instance the responsibility for sub-processes may be delegated, and possibly out-sourced in an e-commerce environment. The system forms a view on who should be asked to do what at each step in a process, and tracks the resulting delegations of process responsibility. The system has been trialed on an emergent process application in a university administrative context.

References

[1] Robert Skinstad, R. "Business process integration through XML". In proceedings XML Europe 2000, Paris, 12-16 June 2000.

[2] Feldman, S. "Technology Trends and Drivers and a Vision of the Future of e-business." In proceedings 4th International Enterprise Distributed Object Computing Conference, September 25-28, 2000, Makuhari, Japan.

[3] Dourish, P. "Using Metalevel Techniques in a Flexible Toolkit for CSCW Applications." ACM Transactions on Computer-Human Interaction, Vol. 5, No. 2, June, 1998, pp. 109—155.

[4] A. P. Sheth, D. Georgakopoulos, S. Joosten, M. Rusinkiewicz, W. Scacchi, J. C. Wileden, and A. L. Wolf. "Report from the NSF workshop on workflow and process automation in information systems." SIGMOD Record, 25(4):55—67, December 1996.

[5] Debenham, J.K. "Three Intelligent Architectures for Business Process Management", in proceedings 12th International Conference on Software Engineering and Knowledge Engineering SEKE2000, Chicago, 6-8 July 2000.

[6] Jain, A.K., Aparicio, M. and Singh, M.P. "Agents for Process Coherence in Virtual Enterprises" in Communications of the ACM, Volume 42, No 3, March 1999, pp62—69.

[7] Debenham, J.K. "Supporting knowledge-driven processes in a multiagent process management system." In proceedings Twentieth International Conference on Knowledge-Based Systems and Applied Artificial Intelligence, ES'2000: Research and Development in Intelligent Systems XV, Cambridge UK, December 2000.

[8] Fischer, L. (Ed) "Workflow Handbook 2001." Future Strategies, 2000.

[9] Debenham, J.K. "Supporting Strategic Process", in proceedings Fifth International Conference on The Practical Application of Intelligent Agents and Multi-Agents PAAM2000, Manchester UK, April 2000.

[10] Debenham, J.K. "Knowledge Engineering: Unifying Knowledge Base and Database Design", Springer-Verlag, 1998

[11] Muth, P., Wodtke, D., Weissenfels, J., Kotz D.A. and Weikum, G. "From Centralized Workflow Specification to Distributed Workflow Execution." In Journal of Intelligent Information Systems (JIIS), Kluwer Academic Publishers, Vol. 10, No. 2, 1998.

[12] Rao, A.S. and Georgeff, M.P. "BDI Agents: From Theory to Practice", in proceedings First International Conference on Multi-Agent Systems (ICMAS-95), San Francisco, USA, pp 312—319.

[13] Durfee, E.H.. "Distributed Problem Solving and Planning" in Weiss, G. (ed). Multi-Agent Systems. The MIT Press: Cambridge, MA.

[14] Finin, F. Labrou, Y., and Mayfield, J. "KQML as an agent communication language." In Jeff Bradshaw (Ed.) Software Agents. MIT Press (1997).

Diagnosis Algorithms for a Symbolically Modeled Manufacturing Process

N. Rakoto-Ravalontsalama

Department of Automatic Control
Ecole des Mines de Nantes
4, rue Alfred Kastler, F-44307 Nantes Cedex 03, France
rakoto@auto.emn.fr

Abstract. The considered manufacturing process is modeled in both numerical and symbolic ways, in a disjoint manner. This paper focuses on the symbolic model of the given Discrete Event System (DES). A new method for diagnosing some symbolic parameters is presented. The proposed diagnosis algorithms are to be performed sequentially. The simulation results show that these algorithms have improved the symbolic part of the process model.

1 Introduction

Fault detection and isolation has received considerable attention in the literature because it is an important task in automatic control and complex systems. Many approaches have been applied to industrial large processes [11] or [2].

It is far from obvious to build up a complete model for a given process. Thus dealing with incomplete knowledge is quite relevant to knowledge representation. Various symbolic (qualitative) approaches have been proposed to cope with such an incomplete knowledge including *sign algebra* which is the most elementary (thus incomplete) way of representation.

Ambiguity is the predominant source of weakness in sign algebra and it has been a major focus of research in qualitative reasoning like FOG [12], O(M) [9], SR1 [5] and ROM(K) [3]. These are some examples of approaches that bring improvements with regard to the classical sign algebra.

This paper focuses on a particular Discrete Event System (DES) which is a manufacturing line. It transforms a raw material into a finished product. The system has a hierarchical structure that are subprocess, workcell and workstation. The process is composed of a set of sequential subprocesses. Each subprocess is made of a number of sequential workcells. Lastly each workcell is a set of sequential workstations. The workstation is the elementary component, typically a machine. It is modeled in both numerical and symbolic way, but in a disjoint manner (no model overlapping).

The aim of the paper is to enhance the symbolic model, especially by detecting and isolating the uncertain (or ambiguous) symbols. For the rest of the paper, uncertain and ambiguous have the same meaning that is the symbolic value $q_?$.

V.N. Alexandrov et al. (Eds.): ICCS 2001, LNCS 2073, pp. 1228–1236, 2001.
© Springer-Verlag Berlin Heidelberg 2001

Then a diagnosis algorithm to deal with such an uncertain information is proposed. This is based on two sequential algorithms that are *coarse diagnosis* followed by *fine diagnosis*.

This paper is organized as follows: Section 2 briefly presents a summary of the previous process model, section 3 proposes the basic principles of the two diagnosis algorithms. A discussion is given in section 4.

2 Preliminaries

The DES is a TV-tube manufacturing system. This has been modeled in order to build a hierarchical control of the whole process. The system has been decomposed using a hierarchical approach that are *subprocess, workcell* and *workstation*. Each component is made of sequential sets of sub-components. A basic specification has been defined for the most elementary element, i.e., the workstation. The basic model is summarized here: for more details see [7], [8], and [14].

A *workstation* is the most elementary machine, typically consisting of one processing tool, position or action. The task performed at this position is generally done by a robot. Firstly, the workstation model takes into account some properties of classical control systems like controllability and observability. Thus distinction has been made between *control* and *uncontrollable* parameters (generally inputs) as well as *observable* and *unobservable* parameters (mostly outputs). Secondly, the model is focused on the product itself, the TV-tube, but also includes some process parameters. Finally, each parameter can be of numerical or symbolic type. These types of specification are given as follows:

- *Numerical (quantitative) behavioral model*: it mostly consists of algebraic relations and ordinary differential equations. This type of model is recognized to be the best one when available. Some examples of significant numerical variables are the tube temperature or its rotation speed at a given machine.
- *Symbolic (qualitative) behavioral model*: it is used when the quantitative specification is not available, either when there is not enough information (most of the cases) or when the numerical information extraction is costly too expensive.

It can be noticed that the symbolic and numerical models are disjoint sets, but the two type of parameters interact. The symbolic specification is mainly used to deal with rough knowledge, which are represented by the following set of symbols $S = \{q_-, q_0, q_+\}$. The meaning of each symbol is as follows: the sign represents the deviation from the nominal value of a given variable. It is then obvious that the *best* process model corresponds to the one where all symbolic variables take their nominal value i.e., the symbolic value q_0. The sign addition and sign multiplication are given in Table 1.

Table 1. Sign addition (left) and sign multiplication (right)

$\oplus$	q_-	q_0	q_+	$q_?$
q_-	q_-	q_-	$q_?$	$q_?$
q_0	q_-	q_0	q_+	$q_?$
q_+	$q_?$	q_+	q_+	$q_?$
$q_?$	$q_?$	$q_?$	$q_?$	$q_?$

$\otimes$	q_-	q_0	q_+	$q_?$
q_-	q_+	q_0	q_-	$q_?$
q_0	q_0	q_0	q_0	q_0
q_+	q_-	q_0	q_+	$q_?$
$q_?$	$q_?$	q_0	$q_?$	$q_?$

This symbolic model seemed to be not enough to model a part of the DES. Thus the classical set S has been extended to a set S'' by adding an *ambiguous* value ($q_?$) and all *alternate* values $S_d = \{(q_-, q_0), (q_0, q_+), (q_-, q_+)\}$. These represent the binary logical OR of two basic symbols. For details on the symbolic specification, see [13] and [14].

- Sign algebra set $S = \{q_-, q_0, q_+\}$, $S' = S \cup \{q_?\}$
- *Alternate value* $(q_a, q_b) = q_a \vee q_b$ (logical-OR of q_a and q_b).
- $S_d = \{(q_-, q_0), (q_0, q_+), (q_-, q_+)\}$
- Universe of calculus: $S'' = S' \cup S_d$

Using the set S'', some basic mappings like `Qplusp(), Plusp() and Distp()` have been defined. The addition operation extended to set S'' is shown in Table 2 and the multiplication operator extended to set S'' is given in Table 3.

Table 2. Sign addition extended to set S''

$\oplus$	q_-	(q_-, q_0)	q_0	(q_0, q_+)	q_+	(q_-, q_+)	$q_?$
q_-	q_-	q_-	q_-	$q_?$	$q_?$	$q_?$	$q_?$
(q_-, q_0)	q_-	(q_-, q_0)	(q_-, q_0)	$q_?$	$q_?$	$q_?$	$q_?$
q_0	q_-	(q_-, q_0)	q_0	(q_0, q_+)	q_+	(q_-, q_+)	$q_?$
(q_0, q_+)	$q_?$	$q_?$	(q_0, q_+)	(q_0, q_+)	q_+	$q_?$	$q_?$
q_+	$q_?$	$q_?$	q_+	q_+	q_+	$q_?$	$q_?$
(q_-, q_+)	$q_?$	$q_?$	(q_-, q_+)	$q_?$	$q_?$	$q_?$	$q_?$
$q_?$	$q_?$	$q_?$	$q_?$	$q_?$	$q_?$	$q_?$	$q_?$

Finally, by combining some basic mappings which are associated with the sign operators $\oplus$ and $\ominus$, some *compound mappings* like (`Plus(), Dist(), Tune()`) have been defined. The basic and compound mappings definitions are specified in Table 4. For example, the compound mapping `Plus()` is a combination of two basic mappings `plusp()` obtained with a subtraction operator. The meaning of each mapping and more details on the symbolic specification can be found in [13] and [14].

Table 3. Sign multiplication extended to set S''

$\otimes$	q_-	(q_-, q_0)	q_0	(q_0, q_+)	q_+	(q_-, q_+)	$q_?$
q_-	q_+	q_0, q_+	q_0	q_-, q_0	q_-	q_-, q_+	$q_?$
(q_-, q_0)	q_0, q_+	(q_0, q_+)	q_0	(q_-, q_0)	(q_-, q_0)	$q_?$	$q_?$
q_0	q_0	q_0	q_0	q_0	q_0	q_0	q_0
(q_0, q_+)	q_-, q_0	q_-, q_0	q_0	(q_0, q_+)	(q_0, q_+)	$q_?$	$q_?$
q_+	q_-	(q_-, q_0)	q_0	(q_0, q_+)	q_+	(q_-, q_+)	$q_?$
(q_-, q_+)	$(q_-, q_+$	$q_?$	q_0	$q_?$	(q_-, q_+)	(q_-, q_+)	$q_?$
$q_?$	$q_?$	$q_?$	q_0	$q_?$	$q_?$	$q_?$	$q_?$

Table 4. Basic and compound symbolic mappings

Mappings	Initial set(s)	Final set	Expression
$\texttt{Qlusp}(s)$	S''	S''	$= \begin{cases} q_0 & \text{if } (s = q_- \text{ or } s = (q_-, q_0) \text{ or } s = q_0), \\ q_+ & \text{if } s = q_+, \\ (q_0, q_+) & \text{otherwise.} \end{cases}$
$\texttt{Plusp}(x, \Delta)$	$R \times R$	S''	$= \begin{cases} q_0 & \text{if } x \le 0, \\ (q_0, q_+) & \text{if } 0 < x \le \Delta, \\ q_+ & \text{if } x > \Delta. \end{cases}$
$\texttt{Distp}(x, \Delta)$	$R \times R$	S''	$= \begin{cases} q_0 & \text{if } x \le 0, \\ q_? & \text{if } 0 < x \le \Delta, \\ (q_-, q_+) & \text{if } x > \Delta. \end{cases}$
$\texttt{Plus}(x, \Delta_1, \Delta_2)$	$R \times R \times R$	S''	$= \texttt{Plusp}(x - \Delta_1, \Delta_2) \ominus \texttt{Plusp}(-x - \Delta_1, \Delta_2)$
$\texttt{Dist}(x, \Delta_1, \Delta_2)$	$R \times R \times R$	S''	$= \texttt{Distp}(x - \Delta_1, \Delta_2) \oplus \texttt{Distp}(-x - \Delta_1, \Delta_2)$
$\texttt{Tune}(x, \Delta_1, \Delta_2)$	$R \times R \times R$	S''	$= \texttt{Plusp}(x - \Delta_1, \Delta_2) \oplus \texttt{Plusp}(-x - \Delta_1, \Delta_2)$

3 Diagnosis

Fault diagnosis generally consists in fault detection and then isolation. In this work, fault is not the "usual" failure that can be found in industrial complex systems but the ambiguous symbolic variable $q_?$.

We recall that the symbolic knowledge is expressed by the sign algebra set, $S_d = \{q_-, q_0, q_+\}$, to which additional symbols have been added. These are alternate symbols, $S_d = \{(q_-, q_0), (q_0, q_+), (q_-, q_+)\}$. Furthermore, once an ambiguous symbol $q_?$ appears, it is propagated to another related variables, due to the sequential structure of the DES. The objective of the diagnostic task is to enhance the symbolic model that is to remove all the ambiguous and all the alternate symbols. Thus the target set after diagnosis is the set of sign algebra, $S_f = S = \{q_-, q_0, q_+\}$. The ideal case would be that this target set is reduced to the single null element $\{q_0\}$.

Therefore, to improve the model i.e., to remove all ambiguous and alternate values, two simple sequential algorithms, *coarse diagnosis* and *fine diagnosis* are proposed. The first algorithm is devoted to the treatment of variables taking the ambiguous value $q_?$ while the second one deals with the alternate symbols (belonging to Set S_d). These algorithms are to be performed sequentially, coarse diagnosis first, followed by the fine one.

3.1 Preliminary Assumptions

Some assumptions have to be made before performing such algorithms:

1. There exists at least one symbolic variable taking the ambiguous value $q_?$ at the beginning of the simulation. Every symbolic variable taking such a value is called an *uncertain* variable.
2. According to the structure of the DES and because of the use of the addition operation, an uncertain variable will always forward its uncertain value to every related variable in the next workstations[1].
3. For each *uncertain variable*, there exists at least one control action in order to change its value; this (these) control variable(s) is (are) located only in the same workstation – not general control variable(s).
4. Each control variable is numerical and is bounded, $u \in [u_{min}, u_{max}]$.
5. Coarse diagnosis as well as fine diagnosis are performed off-line.

3.2 Preliminary Causal Graph

The building of the *hybrid causal graph* (HCG) is also required, before performing the diagnostic task. One HCG is built for each workstation. The causal graph is a set of points (nodes) and a set of arcs (relations) connecting some nodes. A *node* is a real-valued or symbolic variable ($x_i \in R \cup S''$). A *directed arc* represents a relation between the connected variables, from the cause(s) to the effect(s). A *numerical arc* is an arc where both cause(s) and effect(s) are numerical. A

[1] This is not true if the Q-multiplication is used, because $q_? \otimes q_0 = q_0$.

symbolic arc is an arc for which the effect variable(s) is (are) symbolic, the cause(s) being numerical or symbolic.

Existing similar approaches are SDG, Signed Directed Graph introduced by Iri et al. [6], ESDG, Enhanced Signed Directed Graphs proposed by Oyeleye and Kramer [10] and QUAF, Qualitative Analysis of Feedback, proposed by Rose and Kramer [15].

3.3 Coarse Diagnosis

The coarse diagnosis objective is to remove the uncertain variables from the initial set S''. The algorithm consists in the isolation of the uncertain variable, then the tuning of the control variable and performing the simulation. Lastly other uncertain(s) variable(s) are then detected. The target set is the initial set without the ambiguous values.

- Initial set: $S_i = S''$
- Objective: to remove all uncertain variables (ambiguous values $q_?$)
- Target set: $S_f = S'' - \{q_?\}$; Ideal case $S_f = \{q_0\}$.
- Algorithm: isolation, tuning control variable, simulation and detection

More precisely the detailed algorithm to perform such a diagnosis task is given below:

Coarse Diagnosis

1. **Initialization:**
   a. Isolate the first subprocess
   b. Isolate the first workcell
   c. Isolate the first workstation
   d. Isolate the first *uncertain variable* i.e. $q?$.
2. **Isolation of uncertain variable:**
   a. Isolate the next subprocess
   b. Isolate the next workcell
   c. Isolate the next workstation
   d. Isolate the next *uncertain variable* i.e. $q_?$.
3. **Performing control action:**
   a. find appropriate control variable (control tuning)
   b. determine its numerical value
   c. apply control action
4. **Running simulation:**
5. **Detection of (other) uncertain variable(s)**
   a. if analyzed qvar is still uncertain then goto 3b.
   ---- else goto 5b.
   b. if at least one uncertain variable is detected in current workstation then goto 1d.
   ---- else goto 5c.
   c. if at least one uncertain variable is detected in current workcell then goto 2c.
   ---- else goto 5d.

```
d. if at least one uncertain variable is detected in current subprocess
then goto 2b.
---- else goto 5e.
e. if at least one uncertain variable is detected in the process then goto
2a. else end.
```

Some remarks can be made upon this coarse diagnosis algorithm:
- The considered *uncertain variable* is assumed to be an output variable (and not a process one); uncertain *process* variable always gives rise to uncertain *output* variable(s) in the same workstation[2].

- It is assumed that there exists at least one uncertain variable at the beginning of the diagnosis task so that the algorithm begins with isolation instead with detection (unlike standard diagnosis: detection, then isolation)

3.4 Fine Diagnosis

- Initial set: $S_i = S'' - \{q_?\}$
- Objective: to remove all alternate variables (S_d)
- Target set: $S_f = S_i - S_d = S$; Ideal case: $S_f = \{q_0\}$
- Algorithm: isolation, tuning control variable, simulation and detection

The fine diagnosis algorithm is exactly the same algorithm as for coarse diagnosis, but the focus is on alternate variables (S_d) instead of ambiguous ones $\{q_?\}$.

4 Discussion

The numerical model is represented by ordinary differential equations, while the symbolic model is expressed by algebraic sign relations and decision tables with `if-then rules` in Table 1, Table 2 and Table 3. The simulation duration is about 1 second while the effective time is 20 minutes. The simulator writes the output variables in a specific file.

The diagnosis algorithms have been applied to a part of the manufacturing system. This system has strong sequential properties. The simulation has been performed to the *Flowcoat Green* workcell, which is a sequential collection of twelve workstations, FG01 to FG12. The main objective of this workcell is to put a green flowcoat liquid on a blank TV tube. A significant workstation is the fourth one, FG04. This workstation is recognized to be significant because at this location is determined the amount of green flowcoat liquid to be spread which will become the green phosphor lines on the finished TV tube.

The coarse diagnosis has detected 10 ambiguous variables, while the fine algorithm have removed 7 alternate symbols. The two diagnosis algorithms have improved the model in the way that the refined model has a reduced number of uncertain variables, thus reducing the total number of states of the hybrid process. However, the proposed method only deal with the symbolic model, that is only a part of the complete model. And the number of initial ambiguous

[2] This holds since no Q-multiplication is used in the present formalism.

variables is relatively low because the diagnosis algorithms have been applied to a model that has been previously tuned (with other methods).

However, many assumptions have been made. The propagation of the ambiguous symbolic variable $q_?$ is observable since only the sign addition operator is used for the symbolic model. Indeed, if the symbolic model include the use of the multiplication operator, an ambiguous symbolic variable $q_?$ could be transformed to q_0, thus making this *failure* unobservable. Another restrictive assumption is the existence of a numerical control action that can be tuned in order to remove the ambiguous variable. This could be quite difficult to obtain.

5 Concluding Remarks

A new method of diagnosing uncertain parameters has been presented. The system is modeled in a disjoint numerical and symbolic way. The main idea is to detect uncertain (or ambiguous) variables in order to improve the symbolic discrete event model. The advantage of such method is the systematic checking of uncertain variables thanks to the symbolic model of the system and also to the sequential structure of the class of the considered discrete event system.

However the extension of this approach to another class of discrete event systems is not straightforward because many assumptions have been made. For example the existence of a control action that can tune the model in order to remove the ambiguous symbolic variable is not always guaranteed. The type of considered DES is quite a simple one. Fault diagnosis in FlowShop or JobShop systems become more complicated due to the multiplicity of machines thus a multiplicity of possible faults.

On-going work concerns the automatic tuning of control variable using lower and upper bounds, where some process expertise is required. Finally the next step is the automated diagnosis of such algorithms in order to perform the diagnostic task on-line.

References

1. A. Aghasaryan, E. Fabre, A. Benveniste, R. Boubour and C. Jard "A Petri net approach to fault detection and diagnosis in distributed systems: Extending Viterbi algorithm and HMM techniques to Petri nets ". In *Proc. 36th IEEE Conf. on Decision and Control*, Dec. 1997.
2. R. Boubour, C. Jard, A. Aghasaryan, E. Fabre and A. Benveniste. "A Petri net approach to fault detection and diagnosis in distributed systems: Application to telecommunication networks, motivations, and modelling". In *Proc. 36th IEEE Conf. on Decision and Control*, Dec. 1997.
3. P. Dague. "Symbolic reasoning with relative orders of magnitude". In *Proc. 13th Int. Conf. on Artificial Intelligence IJCAI'93*, pp. 1509–1514, Chambery, France, 1993.

4. R. Debouk, S. Lafortune and D. Teneketzis. "Coordinated decentralized protocols for failure diagnosis of discrete event systems". In *Proc. 37th IEEE Conf. on Decision and Control*, Dec. 1998.
5. J. de Kleer and B.C. Williams. "Diagnosing multiple faults". *Artificial Intelligence.*, 32:97–130, 1987.
6. M. Iri, K. Aoki, E. O'Schima and H. Matsuyama. "An algorithm for diagnosis of system failure in the chemical process". *Computer Chemical Engineering*, 3:489–493, 1979.
7. J. Kamerbeek and J.S. Kikkert. "Esprit 2428 IPCES: Process description of prototype 2". Private Communication, Philips-PRL and Philips-TCDC, Eindhoven, The Netherlands, December 1991.
8. J. Kamerbeek. "Generating simulators from causal process knowledge". *Proc. of European Simulation Symposium 1993.* Delft, The Netherlands, October 1993.
9. M.L. Mavrovouniotis and G. Stephanopoulos. "Formal order of magnitude reasoning in process engineering". *Computer Chemical Engineering*, 12:867–880, 1988.
10. O.O. Oyeleye and M.A. Kramer. "Qualitative simulation of chemical process systems: Steady-state analysis". *AIChE Journal*, 34:1441–1454, 1988.
11. P.M. Frank. "Analytical and qualitative model-based fault diagnosis – A survey and some new results". *European Journal of Control*, 2:6–28, 1996.
12. O. Raiman. "Order of magnitude reasoning". *Proc. of AAAI-86*, pp. 100–104, Philadelphia, PA, USA . 1986.
13. N. Rakoto-Ravalontsalama, A. Missier, and J.S. Kikkert. "Qualitative operators and process engineer semantics of uncertainty".In B. Bouchon-Meunier, L. Valverde, and R.R. Yager, editors, *Lecture Notes in Computer Science 682*, IPMU'92 – Advanced Methods in Artificial Intelligence, pp. 284–293. Springer Verlag, 1992.
14. N. Rakoto-Ravalontsalama and J. Aguilar-Martin. "Knowledge-based modelling of a TV-tube manufacturing system" *IFAC Journal of Control Engineering Practice*, 4(1), pp. 117-123, Jan. 1996.
15. P. Rose and M.A. Kramer. "Qualitative analysis of causal feedback". *Proc. of AAAI-91*, Anaheim, CA, USA .1991.
16. M. Sampath, S. Lafortune, and D. Teneketzis. "Active diagnosis of discrete event systems". *IEEE Trans. Automat. Contr.*, 43(7), pp. 908–929, July 1998.

Time-Accurate Turbine Engine Simulation in a Parallel Computing Environment Part II - Software Alpha Test[1]

M. A. Chappell and B. K. Feather

Sverdrup Technology, Inc., AEDC Group
Arnold Engineering Development Center, TN

Abstract. The ability to rapidly execute a high-fidelity turbine engine simulation is essential in automating data validation processes during developmental engine testing. Further, execution of a simulation in real time enables validation of a continuously varying data stream concurrently with the data acquisition task.

Component-level turbine engine simulations provide high-fidelity steady-state and time-accurate transient engine performance computations but are not typically applied in a real-time environment. This paper presents an approach for distributing a component-level simulation task in a parallel computing environment in an effort to achieve real-time operation.

A turbine engine simulation was restructured to operate in a parallel computing environment and tested to quantify the resulting impact on simulation fidelity and execution time. A software acceptance test confirmed the viability of the parallel simulation approach and guided the formulation of refinements. The refinements were incorporated, and the program was reevaluated in terms of repeatability, speedup, and scalability. The approach, software alpha test results, and direction of future work are summarized herein.

Introduction

Turbine engine testing at the Arnold Engineering Development Center (AEDC) is conducted to evaluate engine operation over a wide variety of power conditions and simulated altitude conditions. Thousands of sensors, many producing measurements at rates in excess of one hundred samples per second, are typically installed in the engine and in the test facility to measure aerothermodynamic performance. Consequently, a typical 8-hour test can produce three billion samples of aerothermodynamic performance data. The challenge is to ensure the validity of the data, monitor the condition of the engine, and promptly identify anomalies.

The countless variations of steady-state and transient engine operation and the necessity to distinguish between sensor anomalies and abnormal engine deterioration, combined with the large volume of data, overwhelm the capabilities of traditional data

[1] The research reported herein was performed by the Arnold Engineering Development Center (AEDC), Air Force Material Command. Work and analysis for this research were performed by personnel of Sverdrup Technology, Inc., AEDC Group, technical services contractor for AEDC. Further reproduction is authorized to satisfy needs of the U. S. Government.

V.N. Alexandrov et al. (Eds.): ICCS 2001, LNCS 2073, pp. 1237–1250, 2001.
© Springer-Verlag Berlin Heidelberg 2001

validation methods. Although traditional methods produce meaningful results, they are labor-intensive and time-consuming. Consequently, application of traditional methods is typically restricted to a fraction of the available data, thus diminishing the ability to detect anomalous data and intermittent events.

An approach was developed for a fast, comprehensive, and automated data validation process. The process consists of several analysis and modeling methods which together provide a real-time, model-based data validation tool. The methods include an event detection system, a rule-based expert system with more than 150 checks, and an engine model-based fault detection and diagnostic system (Fig. 1). The engine model-based element of the system relies on a component-level turbine engine model (CLM) which employs fundamental physical laws to relate measurements to each other. The CLM is an industry-accepted method [1-3] for simulating the aerothermodynamic performance of turbine engines but is not typically used when real-time simulation is required. However, the CLM provides the level of accuracy and detail required for data validation and fault diagnosis. Other turbine engine modeling methods are available but may compromise accuracy and detail in order to increase execution speed [4]. Furthermore, the CLM is more easily adapted to the frequent changes that are expected during engine development.

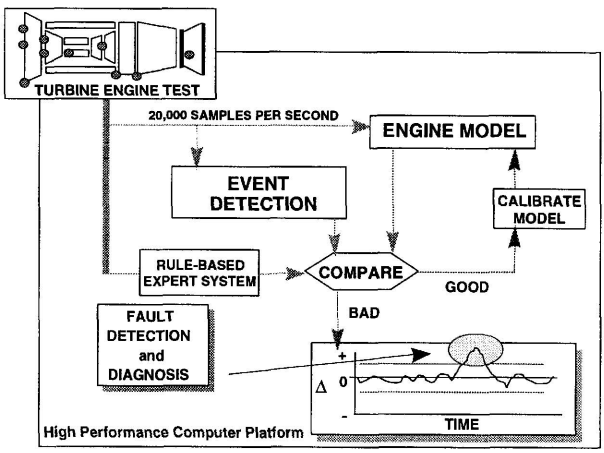


Fig. 1. Real-Time, Model-Based Data Validation System

The CLM must process each data sample at speeds which match or exceed the data-sampling rate since the CLM is only one element of an automated process which monitors a data stream whose measurements vary continuously. Otherwise, a backlog of data samples will grow while waiting for earlier \!data samples to be processed. A "real-time" processing rate is defined as a rate that matches the sampling rate. A requirement also exists for a rate which is faster than the "real-time" rate. This additional requirement addresses a need to process recorded data in a time span which is shorter than the duration of the test. For example, a four-hour data tape, off-loaded from a test vehicle, could be processed in less than four hours.

The challenge is to implement a computing strategy that enables the CLM to execute in real time or faster. Parallel computing was selected as a strategy to provide real-time execution. This strategy also can be scaled up, allowing extension to offline "faster than real time" applications. A temporal decomposition of a time-accurate turbine engine CLM is proposed in which the CLM is replicated on multiple computer processors (CPU) within a high-performance multiple-CPU computer. Each CPU processes a set of synchronized data samples as other sets are processed on additional CPU's. Temporal decomposition was selected rather than spatial decomposition, which is often used for computational fluid dynamic (CFD) applications [5] because previous CLM work suggests that spatial decomposition of a CLM is ineffective [6,7]. Additionally, temporal decomposition of a CLM is more easily distributed to an increasing number of CPU's. Each level of software testing imposes more rigorous standards than the previous, starting with SAT as an entry level assessment of the software and progressing through alpha testing and beta testing.

The objective of this work is to provide an automated, real-time, model-based test data validation computer code. The CLM is one element of an approach to permit comprehensive validation of test data. Each element, including the CLM, is subject to standard software development practices including Software Acceptance Testing (SAT), Alpha testing, and Beta testing prior to a production release.

The subject of this paper is the parallel implementation strategy for the time-accurate CLM. Results of software alpha testing (α-test) and recommended refinements to the strategy are also discussed.

Component-Level Model Description

The component-level turbine engine model (CLM) provides a lumped-volume, component-level, time-accurate simulation applicable to arbitrary engine designs. The CLM is capable of simulating off-operating-line engine operation and uses widely accepted component-matching principles [1-3]. An engine control simulation may also be included as an additional program module.

The CLM combines physical relationships that govern engine operation with empirical relationships that describe individual component performance. The result is an adaptable model for which the effects of changes to engine attributes (e.g., components, configuration, and controls) are incorporated by making changes to the corresponding model attributes (i.e., components, configuration, and controls modules). In addition, the component-matching approach quantifies the effects of changes to the engine attributes, enabling a prediction capability for the data validation and fault diagnosis process.

The CLM is an assembly of components constrained to operate in unison to simulate the engine. An augmented turbofan engine, for example, may include a variable-geometry fan and compressor, combustor, high-pressure and low-pressure turbines, fan bypass duct, mixer, augmentor, and variable exhaust nozzle (Fig. 2). The component models combine thermodynamic process equations with empirically determined component performance relationships to simulate component

performance. An iterative technique is used to satisfy the implicit relationships that constrain the assembly to mass, momentum, and energy conservation principles. Measured engine control variables are used to govern model operation. The effects of rotor acceleration, heat transfer, and off-schedule variable geometry are included, providing a simulation of steady-state and transient engine operation ranging from engine starting conditions to maximum power. The CLM provides accurate simulation of operating temperatures, pressures, mass flows, and rotational speeds for each of the components illustrated in Fig. 2.

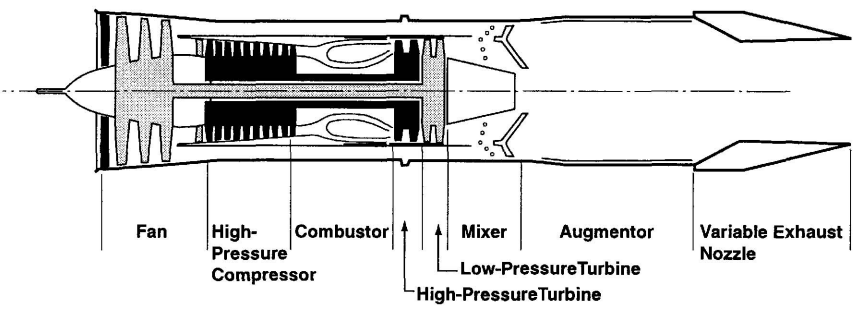


Fig. 2. Augmented Turbofan Engine

Figure 3 illustrates the high level of agreement achievable for model calculations as compared to test measurements for a turbine engine propulsion system. Typical parameters computed by the model include compressor speed and combustor pressure. Agreement briefly exceeds 11 percent but generally is within 5 percent during an in-flight restart sequence, in which the engine experiences maximum power, shutdown, starting, and idle operation.

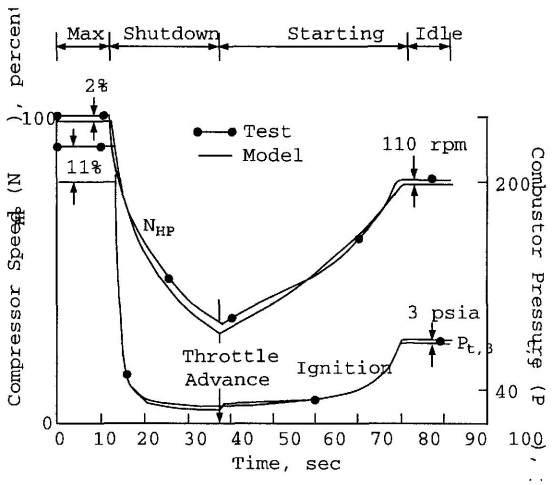


Fig.3. Comparison of Model Calculations to Test Measurements for an Augmented Turbofan Engine

The primary time-dependent effect for the time-accurate CLM is the influence of rotor inertia on engine operation. Heat transfer between the gas path and the engine hardware is a significant but secondary effect. The physical effects of rotor inertia and heat transfer on engine operation are dependent on the time-related history of the affected parameters, and consequently the simulation of these effects is dependent on previously computed values of the affected parameters. The dependence on previous values of the affected parameters imposes a serial character on the simulation computations. The serial character of the physical phenomena and their simulation, coupled with the serial character of numerical integration techniques employed by the CLM, presents a challenge to the parallel computing strategy for the CLM.

CLM Operation in a Test Environment

The CLM, as used in a test environment, is implemented without an engine control simulation module. Avoiding a requirement for a control simulation module eliminates a potential source of error in the data validation process and simplifies the CLM. In the absence of a control simulation, selected test measurements are used as inputs to the CLM to prescribe its forcing functions.

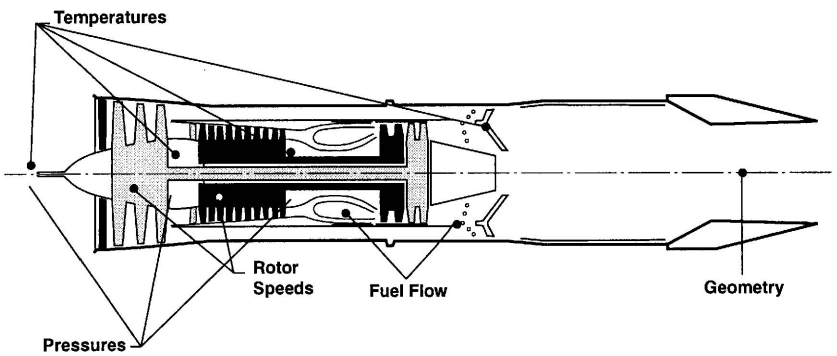


Fig. 4. Test Measurements Used as Inputs to CLM

Typically measurements of the controlled variables, such as fuel flow, are used as inputs along with measurements of the environmental parameters, such as inlet pressure and temperature. Figure 4 illustrates a typical set of pressure, temperature, speed, flow, and variable geometry measurements used as model inputs. Rotor acceleration computed from a time history of rotor speed measurements is also input to the model together with the speed measurement.

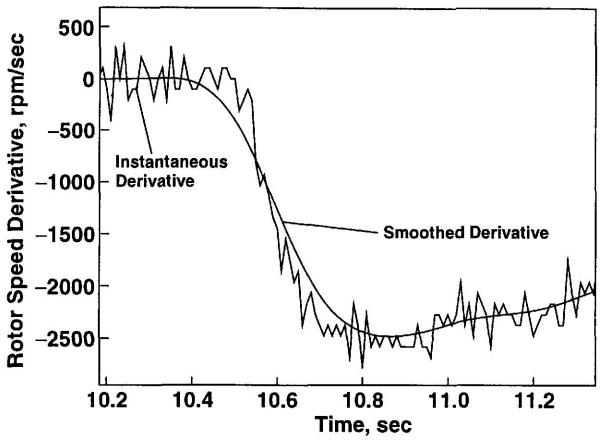


Fig. 5. Engine Deceleration: Rotor Speed Derivative as a Runction of Time

Time-dependent fluctuations in signals for a particular measurement are normal. Although normal, these variations are detrimental to the numerical stability of the model since the model is not designed to respond to rapid (>20 Hz) variations of operating conditions. Unstable model operation reduces fault detection and diagnosis effectiveness and leads to lengthy execution times; therefore, the measurements are smoothed using a time-averaging technique to minimize measurement noise while preserving the mean value of the measurement. Typical variations among fan speed measurements are shown in Figure 5 along with smoothed values used as input to the model.

Parallel CLM Implementation

The time-decomposition of the time-accurate CLM enables a replicated worker approach for a parallel computing implementation of the CLM. The CLM code is replicated on each CPU of a parallel high-performance computer (HPC). Although the results shown herein are from shared memory HPCs, the parallel implementation is also achievable on distributed architectures. All of the worker processors are controlled by a single master processor (Fig. 6). The master processor transmits a user-determined number of data samples to each worker, and the worker processes the samples with its own replication of the model. The alpha version (a-version) of the code ensures that each replicated worker processes a user-selected interval of contiguous samples in contrast to the previous version [8], which processed noncontiguous samples (i.e., each successive sample was processed by a different CPU). The replicated workers receive another interval of samples upon completing each interval; however, the ensuing interval is not contiguous with the previous interval. A coarse-grain worker, discussed below, provides the starting time-dependent boundary conditions for each interval. Additionally, to improve

computational efficiency, all data are loaded into memory before processing is initiated. This sequence precludes disc contention bottlenecks.

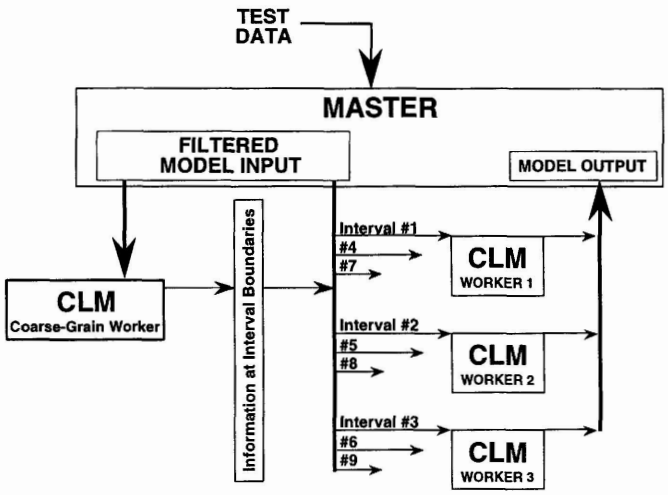


Fig.6. Replicated-Worker Approach for Time-Decomposition of CLM (Three Replicated Workers)

This approach, in which each worker processes an interval of samples, promotes model convergence, reduces the interprocess communication burden, and introduces a challenge in addressing time-dependent effects. Model convergence improves because the model is cognizant of the effect of neighboring samples on the computations. The communication burden is reduced because transmitting many samples with one communication instruction is more effective for the worker than transmitting the same samples with an individual instruction for each sample. The challenge for each worker is to compute the time-dependent effects (e.g., rotor acceleration and heat transfer [8] between nonadjacent intervals. Thus a coarse-grain worker was designed to address the challenge.

The coarse-grain worker accesses the same data samples as the replicated workers but processes only a fraction of them. Also, the coarse-grain worker processes the samples contiguously and computes the time-dependent effects and corresponding boundary conditions for the replicated workers (Fig. 7.). The coarse-grain worker completes its task before the replicated workers need the time-dependent information because the coarse-grain worker processes fewer samples. The coarse-grain worker provides the time-dependent effects to the replicated workers for the starting time of each interval, and the result is fast and accurate computation from the replicated workers.

However, a balance between the granularity of the coarse worker (i.e., the fraction of samples processed) and the scalability of the approach (i.e., the number of replicated workers) is required. As more replicated workers are added to the configuration, the

aggregate rate at which they collectively process the intervals eventually exceeds the rate at which the coarse-grain worker computes the boundary conditions. Further

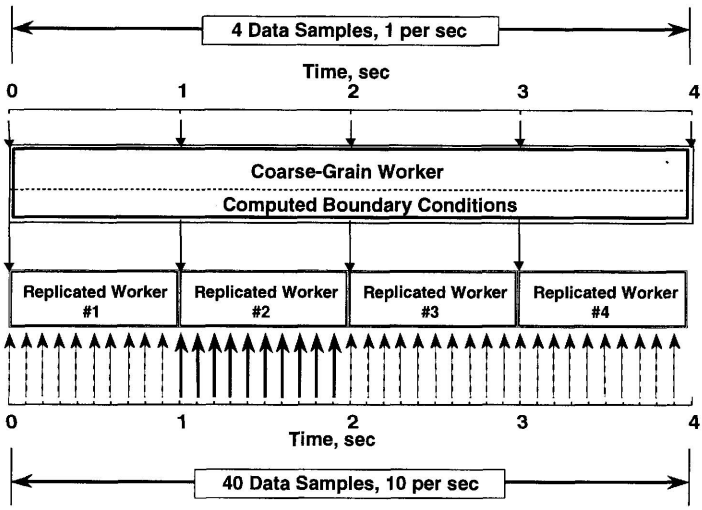


Fig. 7. Example of Relationship Between Coarse-Grain Worker, Replicated Worker, and Data Samples

reduction of the fraction of samples processed by the coarse-grain worker increases its rate but eventually compromises the fidelity of the calculations. Results of the α-test indicate that an acceptable balance is achievable for parallel configurations of up to 32 processors. Parallel configurations of larger than 32 processors have not been tested.

The α-implementation also included a feature to operate the CLM code in a batch mode, thus requiring no interaction from the user. The batch mode enabled extensive automation of software testing. Job scheduling queues on the HPC platforms automatically processed all of the software test cases, testing each case with a wide variety of parallel configurations. Operating within the queuing environment also ensured a stable and consistent environment to conduct repeatable execution time measurements.

Software Alpha-Testing

Two aspects of α-testing were crucial: repeatability verification on multiple CPU configurations, and execution speed measurements. Repeatability verification ensures that the results obtained from a variety of parallel configurations are the same within a tolerance. The execution speed measurements provide an indication of scalability, which is defined as the effectiveness of adding additional CPUs to the process either to reduce execution time of the CLM or to process larger problems.

Repeatability verification, described as the replication of results within a tolerance, was required for all multiple-CPU configurations. Differences in the results between the serial execution (single CPU) and multiple-CPU configurations are attributed primarily to a less accurate starting state for each multiple-CPU interval, a degeneration that results from the temporal decomposition. Acceptable tolerances for the differences were established for individual parameters and were based on the importance of the parameter in the fault detection process, the particular engine maneuver, and the convergence tolerance within the code. Typical comparisons between serial execution, which was considered the baseline, and multi-CPU configurations are shown in Figs. 8 through 13. Acceptable tolerances for α-testing are also contrasted to the higher SAT tolerances in the figures.

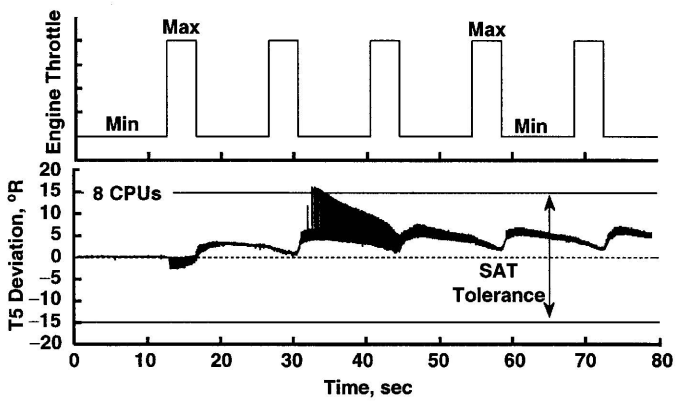


Fig. 8. SAT Turbine Exit Temperature Deviations with Additional CPU's

SAT results [8] are repeated (Figs. 8 and 9) for ease of comparison to α-testing results. Notable improvements in repeatability are apparent in Figs. 10 and 11 relative to SAT results. Figures 8 and 10 show the differences in turbine exit temperature between the serial configuration and an 8-CPU configuration for the SAT and α-implementations, respectively. Turbine exit temperature (T5) was selected for illustration because rotor acceleration and heat transfer affect it significantly. Similarly, Figs. 8 and 10 show the differences in turbine exit pressure between the serial configuration and an 8-CPU configuration for the SAT and α-implementations, respectively. Turbine exit pressure (P56) was selected for illustration because it is sensitive to model convergence tolerances. Time-dependent effects (e.g., rotor acceleration and heat transfer) are included in all cases shown.

The SAT results (Figs. 8 and 9) exhibit a deviation from the baseline during intervals in which the time-dependent effects are significant–in this case, during engine throttle excursions (also shown in Figs. 8 through 11). However, the α-implementation follows the baseline more exactly during the same intervals (Figs. 10 and 11), resulting in differences within one percent, thus verifying the improved repeatability of the α-implementation.

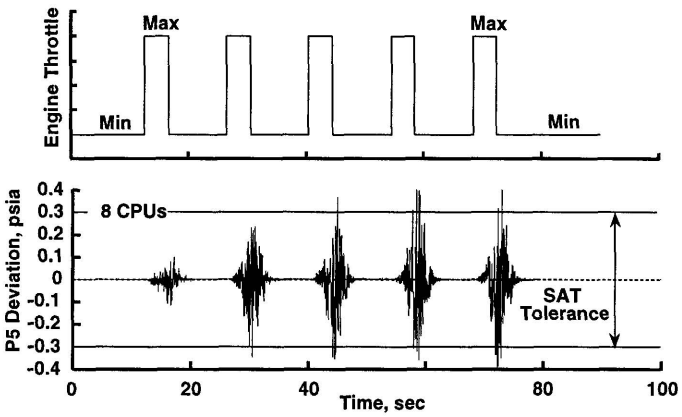


Fig. 9. SAT Turbine Exit Pressure Deviations with Additional CPU's

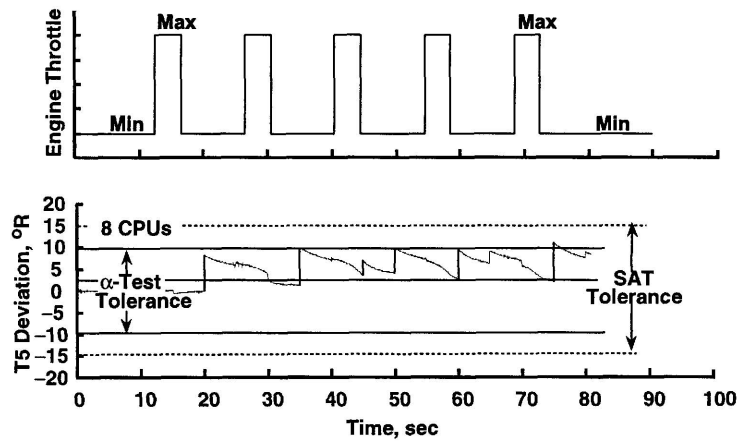


Fig. 10. V-Implementation: Turbine Exit Temperature Deviations with Additional CPU's

Execution speed was also measured during α-testing of the CLM. The same twelve distinct engine maneuvers used during SAT [8] were used during α-testing. Measurements were taken for serial executions of the CLM plus a variety of multiple-CPU configurations ranging from 4 to 32 CPUs on two different massively parallel, high-performance computers (SGI Origin 2000 and HP V2500). Representative results are shown in Figs. 12 and 13. Figure 12 illustrates the speed increase factor relative to the serial execution time (i.e., speedup) achieved by employing additional CPUs. Figure 13 illustrates the speed increase factor relative to the duration of the engine maneuver as an indication of "real-time" capability. Variation in speedup among different engine maneuvers results from the impact of the maneuver on the number of iterations required for model convergence (i.e., severe maneuvers require more iterations). Figure 12 indicates that the speedup of the α-implementation exceeds 50 percent of the maximum theoretical speedup up to the 32-CPU configuration.

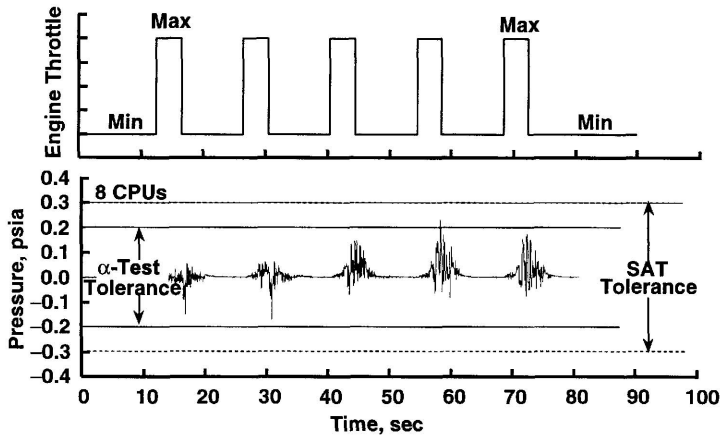


Fig. 11. ∇-Implementation: Turbine Exit Pressure Deviations with Additional CPU's

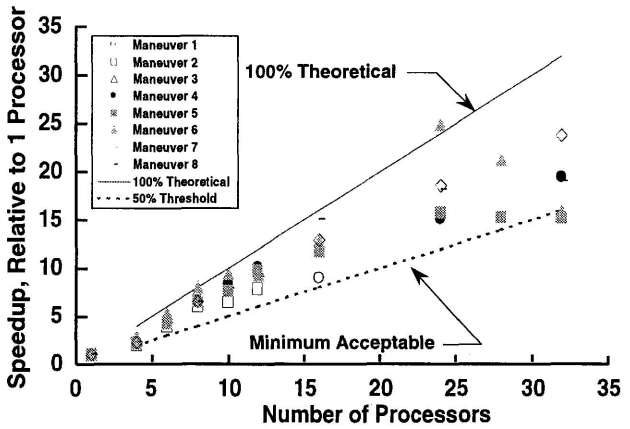


Fig. 12. ∇-Implementation: Parallel Processing Effectiveness in Terms of Speedup

Scalability of the α-implementation was also assessed. Scalability is quantified by the relationship between problem size, or throughput, and the number of CPUs. For the CLM code, scalability is the ability to process more samples within a prescribed length of time by employing more processors. As an example, to achieve 100-percent scalability for a problem size that is two times larger than a selected baseline problem, the code must achieve the same execution time as the baseline with twice as many processors. Longer execution times result in reduced scalability.

Scalability is measured in terms of an overall processing rate which is normalized by the number of CPUs to produce an average processing rate per CPU. Ideally, all parallel configurations will achieve the same normalized processing rate with problems that are sized proportionally with the number of CPUs. Less than ideal performance is realized for a variety of reasons including higher inter-processor communication activity. The normalized processing rate for a smaller problem is

selected as the baseline to which all other rates are compared as an indication of scalability.

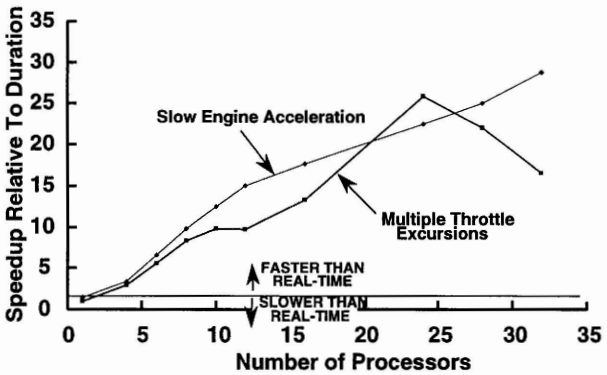


Fig. 13. ∇-Implementation: Speed Increase Factor Relative to Duration of Engine Maneuver

Figure 14 illustrates a comparison of scalability for a larger problem with many CPUs in contrast to a smaller problem with fewer CPUs. The larger problem is composed of four times more samples than the smaller but is otherwise equal in complexity. All values in Fig. 14 are referenced to the baseline case such that a rate equal to that of the baseline is considered ideal. High scalability (96 percent) is realized for the larger problem for 14 worker processors. Although the scalability diminishes as more CPUs are added, it remains above an acceptable level (50 percent) with up to 30 replicated workers.

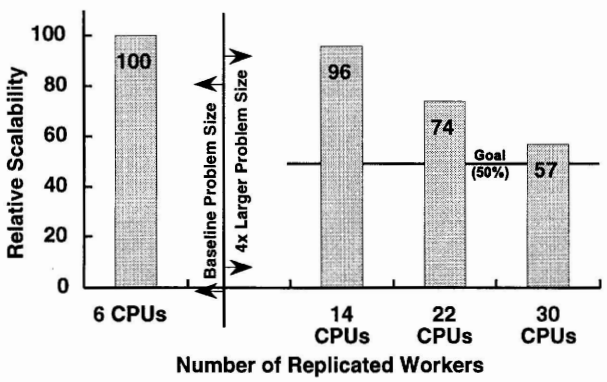


Fig. 14. ∇-Implementation: Relative Scalability
(Average Processing Rate per CPU Relative to Baseline)

Future Work

Three aspects of the code were identified as areas of improvement and the target of future work. The first area focuses on code modularity. Modularity is essential for application of the code in a variety of situations which include both off-line and on-line (i.e., real-time) operation and interactive use and batch operations. The second area focuses on generalization of the CLM code to facilitate application of the code to many different turbine engine designs. Finally, as processors and memory become faster and more expansive, incorporation of more advanced and computationally intensive models [9,10] becomes feasible.

Summary

Component-level turbine engine simulations provide high-fidelity, time-accurate, engine performance computations. An approach for distributing a component-level turbine engine simulation task in a parallel computing environment was implemented and subjected to software alpha-testing (α-test).

Software α-testing confirmed the viability of the component-level model (CLM) code in terms of repeatability verification, speedup, and scalability. Results of the a-testing indicate that the CLM can operate in real time and provide acceptable accuracy to serve as a critical element of a comprehensive, automated data validation process.

References

1. McKinney, J. S., "Simulation of Turbofan Engine, Part I: Description of Method and Balancing Technique," Air Force Aero-Propulsion Laboratory, AFAPC-TR-67-125-PT.-1, 1967.

2. Koenig, R. W. and Fishbach, L. H., "GENENG: A Program for Calculating Design and Off-Design Performance for Turbojet and Turbofan Engines," NASA TN D-6552, 1972.

3. Shapiro, S. R. and Caddy, M. J., "NEPCOMP: The Navy Engine Performance Program," American Society of Mechanical Engineers, 74-GT-83, March 1974.

4. *Real-Time Modeling Methods for Gas Turbine Engine Performance*, Aerospace Information Report, AIR4548, Society of Automotive Engineers, Warrendale, PA, December 1995.

5. Steger, J. L., et al., "A Chimera Grid Scheme," American Society of Mechanical Engineers, Proceedings of the Applied Mechanics, Bioengineering, and Fluids Engineering Conference, Houston, TX, June 20-22, 1983.

6. *Parallel Processor Engine Model Program*, NASA-CR-174641, January 1984.

7. Biegl, C., et al., "Automated, Real-Time Validation of Turbine Engine Test Data Using Explicit Parallelization on an Eight-Processor Pentium Platform," XIII International Symposium on Air-Breathing Engines, ISABE 97-7143, September 1997.

8. Chappell, M. A., et al., "Time-Accurate Turbine Engine Simulation in a Parallel Computing Environment, Part I: Software Acceptance Test," Second Annual International Test and Evaluation Association High Performance Computing Workshop, Edgewood, MD, June 1999.

9. Hale, A. A., et al., "Turbine Engine Analysis Compressor Code: TEACC -- Part II: Multi-Stage Compressors and Inlet Distortion," American Institute of Aeronautics and Astronautics, AIAA-99-3214, 1999.

10. Lytle, J. K., "The Numerical Propulsion System Simulation: A Multidisciplinary Design System for Aerospace Vehicles," XIV International Symposium on Air-Breathing Engines, ISABE 99-7111, September 1999.

Finding Steady State of Safety Systems Using the Monte Carlo Method

Ray Gallagher
rayg@csc.liv.ac.uk

Department of Computer Science, University of Liverpool
Chadwick Building, Peach Street,
Liverpool, UK

Abstract ***In this paper we consider finding the steady state of a safety system. We present parallel Monte Carlo Algorithms for solving eigenvalue problems which we employ to find the steady state.***

The algorithms run on a cluster of workstations under MPICH. Examples would be drawn from safety systems such as controllers etc.

Keywords: Monte Carlo Algorithms, Markov Chain, Parallel Algorithms

1 Introduction

Markov modelling techniques have been used extensively in reliability engineering, particularly with regard to repairable systems that are typical in an industrial environment. These systems offer many advantages in terms of system availability and safety. We will show that the problem of reliability of the system considered can be reduced to finding the dominant eigenvalue(s). Thus, generally, the problem transposes to a linear algebra problem.

Let us now consider Monte Carlo menthods for Linear Algebra. Monte Carlo methods give statistical estimates for the functional of the solution by performing random sampling of a certain random variable whose mathematical expectation is the desired functional. They can be implemented on parallel machines efficiently due to their inherent parallelism and loose data dependencies. Using powerful parallel computers it is possible to apply Monte Carlo methods for evaluating large-scale irregular problems, which are sometimes difficult to solve by well-known numerical methods.

Let J be any functional that we estimate by the Monte Carlo method; θ_N be the estimator, where N is the number of trials. The probable error for the usual Monte Carlo method [7] is defined as parameter r_N for which $Pr\{|J-\theta_N| \geq r_N\} = 1/2 = Pr\{|J-\theta_N| \leq r_N\}$. If the standard deviation is bounded, i.e. $D(\theta_N) < \infty$, the normal convergence in the central limit theorem holds, so we have

$$r_N \approx 0.6745 D(\theta_N) N^{-1/2}. \tag{1}$$

V.N. Alexandrov et al. (Eds.): ICCS 2001, LNCS 2073, pp. 1253–1261, 2001.
© Springer-Verlag Berlin Heidelberg 2001

In this paper we present Monte Carlo algorithms for evaluating the dominant eigenvalue of large real sparse matrices and their parallel implementation on a cluster of workstations using MPICH and how they can be used to find the steady state of systems. These algorithms use the idea of the Power method combined with Monte Carlo iterations by the given matrix and the inverse matrix correspondingly. In [8], [7], [5], [6] one can find Monte Carlo methods for evaluation of the dominant (maximal by modulus) eigenvalue of an integral operator. In [9], [10] Monte Carlo algorithms for evaluating the smallest eigenvalue of real symmetric matrices are proposed. Here we generalize the problem. Finding dominant eigenvalues by parallel Monte Carlo methods can be found in [10],[11]

2 Background

2.1 Algebraic Approach to Calculating Steady State

Assume we are given a transition matrix P. (In most cases of safety systems such matrices are sparse.) Applying an algebraic approach to obtain the steady state transition matrix we have $P^{n+1} = PP^n = P^nP$. Let $K = \lim P^n$ as n tends to infinity, assuming the limit exists. Then K is the solution of $K = PK$ and $K = KP$.

Let us consider evaluating P^n. Consider a Markov Chain with a finite number of states k, where the transition matrix P is a $k \times k$ array (as above). We assume that the eigenvalues of P are λ_i, $i = 1, 2, ..., n$ and $Px = \lambda x$ has non-zero solutions x if, and only if, λ is an eigenvalue of P. Hence, we can find the corresponding right-hand eigenvectors x_i of P. The matrix P can be decomposed into a similar system of matrices comprising its eigenvalues and eigenvectors $P = B\Lambda B^{-1}$. Where Λ is a diagonal matrix of its eigenvalues $\lambda_1, \lambda_2 ... \lambda_n$, and where the matrix B is formed from the corresponding eigenvectors. It follows immediately that $P^n = (B\Lambda B^{-1})^n = B\Lambda^n B^{-1}$.

Therefore, we can calculate the steady state of the system. We will use the Monte Carlo method to calculate the eigenvalues and the method is described below.

2.2 Power Method.

Suppose $A \in R^{n\times n}$ is diagonalizable, $X^{-1}AX = diag(\lambda_1, \ldots, \lambda_n)$, $X = (x_1, \ldots, x_n)$, and $|\lambda_1| > |\lambda_2| \geq \ldots \geq |\lambda_n|$. Given $f^{(0)} \in C^n$, the *power method* ([2]) produces a sequence of vectors $f^{(k)}$ as follows:

$z^{(k)} = Af^{(k-1)}$),
$f^{(k)} = z^{(k)}/||z^{(k)}||_2$,
$\lambda^{(k)} = [f^{(k)}]^H Af^{(k)}$, $k = 1, 2, \ldots$.

Except for special starting points, the iterations converge to an eigenvector corresponding to the eigenvalue of A with largest magnitude (*dominant eigenvalue*) with rate of convergence:

$$|\lambda_1 - \lambda^{(k)}| = O\left(\left|\frac{\lambda_2}{\lambda_1}\right|^k\right). \tag{2}$$

Consider the case when we want to compute the smallest eigenvalue. To handle this case and others, the power method is altered in the following way: The iteration matrix A is replaced by B, where A and B have the same eigenvectors, but different eigenvalues. Letting σ denote a scalar, then the three common choices for B are: $B = A - \sigma I$ which is called the *shifted power method*, $B = A^{-1}$ which is called the *inverse power method*, and $B = (A - \sigma I)^{-1}$ which is called the *inverse shifted power method*.

Table 1. Relationship between eigenvalues of A and B

B	Eigenvalue of B	Eigenvalue of A
A^{-1}	$\frac{1}{\lambda_A}$	$\frac{1}{\lambda_B}$
$A - \sigma I$	$\lambda_A - \sigma$	$\lambda_B + \sigma$
$(A - \sigma I)^{-1}$	$\frac{1}{\lambda_A - \sigma}$	$\sigma + \frac{1}{\lambda_B}$

Computational Complexity: Having k iterations, the number of arithmetic operations in the Power method is $O(4kn^2 + 3kn)$, so the Power method is not suitable for large sparse matrices. In order to reduce the computational complexity we propose a Power method with Monte Carlo iterations.

3 Monte Carlo Algorithms

3.1 Monte Carlo Iterations

Consider a matrix $A = \{a_{ij}\}_{i,j=1}^n$, $A \in R^{n \times n}$, and vectors $f = (f_1, \ldots, f_n)^t \in R^{n \times 1}$ and $h = (h_1, \ldots, h_n)^t \in R^{n \times 1}$. The algebraic transformation $Af \in$

$R^{n\times 1}$ is called *iteration* and plays a fundamental role in iterative Monte Carlo methods.

Consider the following Markov chain:

$$k_0 \to k_1 \to \ldots \to k_i, \tag{3}$$

where $k_j = 1, 2, \ldots, n \quad \text{for } j = 1, \ldots, i$ are natural numbers. The rules for constructing the chain (3) are:

$$Pr(k_0 = \alpha) = \frac{|h_\alpha|}{\sum_{\alpha=1}^{n} |h_\alpha|},$$

$$Pr(k_j = \beta | k_{j-1} = \alpha) = \frac{|a_{\alpha\beta}|}{\sum_{\beta=1}^{n} |a_{\alpha\beta}|}, \quad \alpha = 1, \ldots, n.$$

Such a choice of the initial density vector and the transition density matrix leads to *almost optimal* Monte Carlo algorithms for matrix computations.

Define the random variables W_j using the following recursion formula:

$$W_0 = \frac{h_{k_0}}{p_{k_0}}, \quad W_j = W_{j-1}\frac{a_{k_{j-1}k_j}}{p_{k_{j-1}k_j}}, \quad j = 1, \ldots, i. \tag{4}$$

3.2 Direct Monte Carlo Algorithm

The dominant eigenvalue can be obtained using the iteration process mentioned in the introduction:

$$\lambda_{max} = lim_{i\to\infty}\frac{(h, A^i f)}{(h, A^{i-1} f)},$$

where we calculate scalar products having in mind [7], [8], [4] that

$$(h, A^i f) = E\{W_i f_{k_i}\}, \quad i = 1, 2, \ldots. \tag{5}$$

Thus we have

$$\lambda_{max} \approx \frac{E\{W_i f_{k_i}\}}{E\{W_{i-1} f_{k_{i-1}}\}}$$

3.3 Balancing of Errors

There are two kind of errors in the Power method with Monte Carlo iterations:

- systematic error (from the Power method, (2)):

$$O\left(\left|\frac{\mu_2}{\mu_1}\right|^k\right),$$

  where $\mu_i = \lambda_i$ if $B = A$, $\mu_i = \frac{1}{\lambda_i}$ if $B = A^{-1}$, $\mu_i = \lambda_i - \sigma$ if $B = A - \sigma I$, $\mu_i = \frac{1}{\lambda_i - \sigma}$ if $B = (A - \sigma I)^{-1}$ and λ_i and μ_i are the eigenvalues of A and B correspondingly;
- stochastic error (because we calculate mathematical expectations approximately, (1)):

$$O(D(\theta_N)N^{-1/2})$$

To obtain good results the stochastic error must be approximately equal to the systematic one. It is not necessary to use a *large* number of realizations N in order to have a *small* stochastic error if the systematic error is *large*.

3.4 Computational Complexity

Monte Carlo algorithms can be used to reduce the number of required operations to find the dominant eigenvalue. Dimov and Karaivanova have shown the mathematical expectation of the total number of operations for the Resolvent MC Method ([9], [10] is:

$$ET_1(RMC) \approx 2\tau \left[(k + \gamma_A)l_A + \frac{1}{2}dl_L\right] lN + 2\tau n(1 + d), \qquad (6)$$

where l is the number of moves in every Markov chain, N is the number of Markov chains, d is the mean value of the number of non-zero elements per row, γ_A is the number of arithmetic operations for calculating the random variable (in our code-realization of the algorithm $\gamma_A = 6$), l_A and l_L are arithmetic and logical sub-operations in one move of the Markov chain, k is the number of arithmetic operations for generating the random number (k is equal to 2 or 3).

The main term of (6) does not depend on n, the size of the matrix. This means that the time required for calculating the eigenvalue by RMC is practically independent of n. The parameters l and N depend on the spectrum of the matrix, but do not depend on its size n. The above mentioned result was confirmed for a wide range of matrices during the realized numerical experiments [10],[11].

4 Numerical Tests

The numerical tests are made on a cluster of 48 Hewlett Packard 900 series 700 Unix workstations under MPICH (version 1.1). The workstations are networked via 10Mb switched ethernet segments and each workstation has at least 64Mb RAM and run at least 60 MIPS. Each processor executes the same program for N/p number of trajectories, i.e. it computes N/p independent realizations of the random variable (here p is the number of processors). At the end the host processor collects the results of all realizations and computes the desired value. The computational time does not include the time for initial loading of the matrix because we consider our problem as part of a bigger problem (for example, *spectral portraits of matrices*) and suppose that every processor constructs it.

The test matrices are sparse and stored in *packed row format* (i.e. only nonzero elements). The results for average time and efficiency are given in table 2 and look promising. The relative accuracy is 10^{-3} [11].

We consider the *parallel efficiency* E as a measure that characterizes the quality of the proposed algorithms. We use the following definition:

$$E(X) = \frac{ET_1(X)}{pET_p(X)},$$

where X is a Monte Carlo algorithm, $ET_i(X)$ is the expected value of the computational time for implementation of the algorithm X on a system of i processors.

5 Discussion

5.1 A Numerical Method

An alternative numerical method to calculate the steady states is proposed by Goble [3].

Let us consider as an example a 1oo2 Duplex Control System which has 5 states. We apply a Markov model solution with a numerical approach (Steady State 1oo2 System) using failure rate values given in the transition matrix [1].

The limiting state probabilities for the system of eventual failure into state 3 and state 4 may be calculated as follows: The matrix P can be decomposed to form four sub-matrices of the form shown below:

$$P = \begin{bmatrix} Q & R \\ null & I \end{bmatrix}$$

Table 2. Direct Monte Carlo Algorithm using MPI (number of trajectories - 100000).

	matrix n=128	matrix n=1024	matrix n =2000
1 pr T (ms)	34	111	167
2 pr T (ms)	17	56	83
E	1	0.99	1
3 pr T (ms)	11	37	56
E	1.03	1	1
4 pr T (ms)	8	27	42
E	1.06	1.003	1
5 pr T(ms)	7	21	35
E	0.97	1.06	0.96

Where the upper-left sub-matrix Q is used to calculate the "Mean Time to Failure" or MTTF. The lower left matrix is a null matrix that contains only zeros and the lower right matrix is an "Identity" matrix. The upper-right matrix is called the R matrix and relates transition from the transient states to the absorbing states.

$$Q = \begin{bmatrix} 0.99999005 & 0.0000039537 & 0.0000006363 \\ 0.125 & 0.87499450 & 0.0 \\ 0.0 & 0.0 & 0.9999945 \end{bmatrix}$$

$$R = \begin{bmatrix} 0.000005105 & 0.000000255 \\ 0.00000295 & 0.00000255 \\ 0.00000295 & 0.00000255 \end{bmatrix}$$

The matrix P is truncated to obtain a matrix Q formed by crossing-out the rows and columns of the absorbing states of transition matrix P. The Q matrix is then subtracted from the identity matrix I to form matrix $I - Q$. The above $I - Q$ matrix is then inverted to obtain the matrix N.

$$N = \begin{bmatrix} 166764.67 & 5.27 & 19293.16 \\ 166757.33 & 13.27 & 19292.31 \\ 0.0 & 0.0 & 181818.18 \end{bmatrix}$$

Multiplying the N matrix by the R matrix leads to the limiting state probabilities for the system. (a point where $P^{n+1} = P^n$ and the transition probabilities will not change for higher powers of matrix P).

$$NR = \begin{bmatrix} 0.908264011 & 0.091735989 \\ 0.908247648 & 0.091752352 \\ 0.536363636 & 0.463636364 \end{bmatrix}$$

The values in the elements of the matrix NR relate to the R submatrix in P as shown below.

Thus, if the system starts in state "0" it will end in the state given by [1 0 0 0 0] * P = [0 0 0 0.908264011 0.091735989]

Similarly, starting from state 1, the multiplying row vector would be [0 1 0 0 0] resulting in row 2 of matrix NR, [0 0 0 0.908247648 0.091752352], and starting from state 3, the multiplying row vector would be [0 0 1 0 0] resulting in row 3 of matrix NR, [0 0 0 0.536363636 0.463636364] .

Hence, the limiting state probability that the system fails with outputs de-energised (state 3) is 0.908264011. The limiting state probability the system fails with outputs energised "fail-to-danger" (state 4) is 0.091735989.

5.2 Comparison of Methods

Whereas, the Monte Carlo solution for the eigenvalues of the system is applicable to large scale problems, methods such as Goble's become impracticable. As we have seen the parallel Monte Carlo method can be used to find several of the maximal eigenvalues efficiently or, if necessary, all eigenvalues. In the case where we seek all the eigenvalues then we can use the spectral resolution representation, and we are able to calculate the relevant P^n and thus the steady state of the system. The method is advantageous in that it is the magnitude of the underlying eigenvalues (in particular the dominant eigenvalue(s)) that govern both the manner and the rate of degeneracy of these systems to their limiting state probabilities.

6 Conclusion

Parallel Monte Carlo algorithms for calculating eigenvalues are outlined. The application of these methods for the calculation of the steady states of safety systems is considered.

They can be applied for well balanced matrices (which have nearly equal sums of elements per row) in order to provide good accuracy. We propose to use them when it is required to calculate the dominant eigenvalue or several of the largest eigenvalues of very large sparse matrices. This is because the computational time is almost independent of the dimension of the matrix and also their parallel efficiency is superlinear.

7 Acknowledgements

This work was partially supported by the Health and Safety Executive, UK. Thanks to V. N. Alexandrov for his fruitful discussion and comments.

References

1. R. Gallagher, , MSc Dissertation, Liverpool, 1986.
2. G. H. Golub, Ch. F. Van Loon, *Matrix Computations*, **The Johns Hopkins Univ. Press**, Baltimore and London, 1996.
3. W. M. Goble, *Evaluating Conttrol Systems Reliability*, **Instrumant Society of America**, 1992 pp 235-270.
4. J.H. Halton, *Sequential Monte Carlo Techniques for the Solution of Linear Systems*, **TR 92-033**, University of North Carolina at Chapel Hill, Department of Computer Science, 46 pp., 1992.
5. G.A. Mikhailov, *A new Monte Carlo algorithm for estimating the maximum eigenvalue of an integral operator*, **Docl. Acad. Nauk SSSR, 191**, No 5 (1970), pp. 993 – 996.
6. G.A. Mikhailov, Optimization of the "weight" Monte Carlo methods (Nauka, Moscow, 1987).
7. I.M. Sobol, *Monte Carlo numerical methods*, **Nauka**, Moscow, 1973.
8. V.S.Vladimirov, *On the application of the Monte Carlo method to the finding of the least eigenvalue, and the corresponding eigenfunction*, of a linear integral equation, in Russian: **Teoriya Veroyatnostej i Yeye Primenenie, 1**, No 1 (1956),pp. 113 – 130.
9. I. Dimov, A. Karaivanova and P. Yordanova, *Monte Carlo Algorithms for calculating eigenvalues*, **Springer Lectur Notes in Statistics, v.127** (1998) (H. Niederreiter, P. Hellekalek, G. Larcher and P. Zinterhof, Eds)), pp.205-220.
10. I. Dimov, A. Karaivanova *Parallel computations of eigenvalues based on a Monte Carlo approach*, **Journal of MC Methods and Appl.**, 1998 (to appear).
11. I. Dimov, A. Karaivanova, V. N. Aleksandrov *Performance Analysis of Monte Carlo Algorithms for Eigenvalue Problem*, **ILAS Conference**, Barcelona, 1999.
12. Megson, G., V. Aleksandrov, I. Dimov, *Systolic Matrix Inversion Using a Monte Carlo Method*, **Journal of Parallel Algorithms and Applications** , 3, No 1 (1994), pp. 311-330.

Parallel High-Dimensional Integration: Quasi-Monte Carlo versus Adaptive Cubature Rules

Rudolf Schürer

Department of Scientific Computing, University of Salzburg, AUSTRIA

Abstract. Parallel algorithms for the approximation of a multi-dimensional integral over an hyper-rectangular region are discussed. Algorithms based on quasi-Monte Carlo techniques are compared with adaptive algorithms, and scalable parallel versions of both algorithms are presented. Special care has been taken to point out the role of the cubature formulas the adaptive algorithms are based on, and different cubature formulas and their impact on the performance of the algorithm are evaluated. Tests are performed for the sequential and parallel algorithms using Genz's test function package.

1 Introduction

We consider the problem of estimating an approximation $\mathrm{Q}f$ for the multi-variate integral

$$\mathrm{I}f := \int_{C_s} f(\boldsymbol{x})\, d\boldsymbol{x}$$

for a given function $f : C_s \to \mathbb{R}$, where C_s denotes an s-dimensional hyper-rectangular region $[r_1, t_1] \times \cdots \times [r_s, t_s] \subset \mathbb{R}^s$. Common methods to tackle this problem on (parallel) computer systems are presented in [1,2].

Numerical integration in high dimensions is usually considered a domain of Monte Carlo and quasi-Monte Carlo techniques. This paper will show that adaptive algorithms can be preferable for dimensions as high as $s = 40$.

2 Algorithms

2.1 Quasi-Monte Carlo Integration

Quasi-Monte Carlo methods are the standard technique for high-dimensional numerical integration and have been successfully applied to integration problems in dimensions beyond $s = 300$.

In this implementation a quasi-Monte Carlo algorithm based on Sobol's (t, s)-sequence [3,4] is used. The sequence used in particular is an $(s-1)$-dimensional sequence using Gray code order to speed up generation as described in [5]. The first point of the sequence (the corner $(0, \ldots, 0)$ of the unit cube) is skipped and

V.N. Alexandrov et al. (Eds.): ICCS 2001, LNCS 2073, pp. 1262–1271, 2001.
© Springer-Verlag Berlin Heidelberg 2001

a (t, m, s)-net of dimension s is constructed by adding an additional equidistributed coordinate as described in [3].

This algorithm can be parallelized easily: The net is split into equal sized blocks, with each processing node taking care of one of them. This can be implemented efficiently, because Sobol's sequence allows fast jumping to arbitrary positions. The only communication required is the final gathering of the estimates calculated by each node.

2.2 Adaptive Algorithm

The key concept of adaptive integration is to apply a basic cubature rule successively to smaller subregions of the original integration domain. The selection of these subregions adapts to "difficult" areas in the integration domain by refining subregions with large estimated errors.

The basic sequential algorithm can be outlined as follows:

1. The basic rule is applied to the whole integration domain to estimate the integral and the error of this approximation.
2. The region is stored in the region collection.
3. The region with the largest estimated error is taken from the region collection.
4. This region is split into two subregions.
5. Estimations for result and error for the new subregions are calculated.
6. Both regions are stored in the region collection.
7. If not done, goto step 3

The loop can be terminated either when a certain absolute or relative error is reached or when the number of integrand evaluations exceeds an upper bound.

Parallelization. The most straight-forward parallelization of this algorithm uses a dedicated manager node maintaining the region collection and serving all other nodes with regions for refinement (see e. g. [6,7]). However, as was shown in [8], this approach scales badly even for moderate numbers of processing nodes.

To improve scalability, all global communication has to be removed. This implies that there can not be a dedicated manager node. To balance workload, some communication between processing nodes is required. However, communication has to be restricted to a small number of nodes in the neighborhood of each processing node.

The basic idea of this algorithm is that each node executes the sequential algorithm on a subset of subregions of the initial integration domain. It uses its own (local) region collection to split regions into subregions and to adapt to difficulties found there. The union of all subregions maintained by all processing nodes is always equal to the whole integration domain. So the final result can be obtained easily by summing up the results from all processing nodes.

If this algorithm is used without further load balancing, eventually most of the processing nodes will work on irrelevant refinements on regions with low

(global) estimated errors, while only a few nodes tackle "bad" regions. To avoid this problem, regions with large estimated errors have to be redistributed evenly among processing nodes.

To accomplish this, the nodes are arranged in a G-dimensional periodical mesh. If the number k of processing nodes is a power of 2, a hypercube with dimension $G = \log_2 k$ provides the optimal topology.

After a certain number of refinement steps is performed, each node contacts its direct neighbor in the first of G possible directions and exchanges information about the total estimated error in its local region collection. The node with the larger error sends its worst regions to the other node to balance the total error in both region collections. When redistribution takes place again, it is performed for the next direction. After G redistribution steps, each node has exchanged regions with all its direct neighbors, and a bad region that was sent during the first redistribution step may have propagated to any other node in the mesh by this time. This ensures that bad regions are distributed evenly among all processing nodes.

The basic idea of this algorithm was first published in [9] and was further developed in [10,11,12].

3 Cubature Rules

Adaptive algorithms are based on cubature rules with error estimation, which are defined by two terms

$$\mathrm{Q}^{(1)} f := \sum_{i=1}^{n^{(1)}} w_i^{(1)} f(\boldsymbol{x}_i^{(1)}) \quad \text{and} \quad \mathrm{Q}^{(2)} f := \sum_{i=1}^{n^{(2)}} w_i^{(2)} f(\boldsymbol{x}_i^{(2)}) \ ,$$

with $\mathrm{Q}f := \mathrm{Q}^{(1)} f$ being an approximation for $\mathrm{I}f$, and $\mathrm{E}f := \left|\mathrm{Q}^{(1)} f - \mathrm{Q}^{(2)} f\right|$ being an estimated error bound for the integration error $|\mathrm{I}f - \mathrm{Q}f|$.

Usually interpolatory rules are used, which means that $\mathrm{Q}p$ is exact for all (multivariate) polynomials p up to a certain degree d, i. e. $\mathrm{I}p = \mathrm{Q}p$ for all $p \in \mathbb{P}_d^s$. $\mathrm{Q}^{(2)}$ is usually a cubature rule with a degree less than the degree of $\mathrm{Q}^{(1)}$ and requiring significantly less integrand evaluations, i. e. $n^{(2)} \ll n^{(1)}$. In some cases, the abscissas of $\mathrm{Q}^{(2)}$ are even a subset of the abscissas of $\mathrm{Q}^{(1)}$. For these *embedded rules* no extra integrand evaluations are required to estimate the integration error.

Most empirical evaluations of adaptive integration routines focus on the comparison of different adaptive algorithms, but little is known about how the underlying cubature rules affect the performance of the algorithm. After an initial evaluation of 9 basic cubature rules, the adaptive algorithm described in the previous section was evaluated based on four different cubature rules with error estimation, leading to significantly different results.

3.1 Evaluated Rules

Table 1 lists the basic cubature rules used, together with their degree, their number of abscissas, their sum of the absolute weights $\sum_{i=1}^{n} |w_i|$, which serves as a quality parameter and should be as low as possible, and a reference to the literature.

Table 1. Basic cubature rules

Name	Degree	n	$\sum_{i=1}^{n} \lvert w_i \rvert$	Reference
Octahedron	3	$O(s)$	1	[13]
Hammer & Stroud	5	$O(s^2)$	$0.62s^2 + O(s)$	[14]
Stroud	5	$O(s^2)$	$1.4s^2 + O(s)$	[15]
Phillips	7	$O(s^3)$	$0.23s^3 + O(s^2)$	[16]
Stenger	9	$O(s^4)$	$1.2^4 + O(s^3)$	[17,18]
Genz & Malik	7	$O(2^s)$	$0.041s^2 + O(s)$	[19]

Table 2 lists the four cubature formula pairs that were actually used by the integration algorithm. Formula 7-5-5 is a special case, because it uses 2 additional basic rules $\mathrm{Q}^{(2)}$ and $\overline{\mathrm{Q}^{(2)}}$ for error estimation: $\mathrm{E}f$ is calculated by the formula

$$\mathrm{E}f := \max\left\{\left|\mathrm{Q}^{(1)}f - \mathrm{Q}^{(2)}f\right|, \left|\mathrm{Q}^{(1)}f - \overline{\mathrm{Q}^{(2)}}f\right|\right\} \quad .$$

As we will see, this construction leads to superior results for discontinuous functions, which comes at little additional cost in high dimensions, because there the total number of abscissas is dominated by the nodes of $\mathrm{Q}^{(1)}$.

Table 2. Cubature rules with error estimation

Name	$\mathrm{Q}^{(1)}$	$\mathrm{Q}^{(2)}$
5-3	Hammer & Stroud	Octahedron
7-5-5	Phillips	Hammer & Stroud, Stroud
9-7	Stenger	Phillips
7-5	Genz & Malik	

Genz & Malik does already contain an embedded fifth degree rule for error estimation, so no additional basic rule is required for 7-5. This formula has often been used throughout the literature, primarily due to its small number of abscissas for $s \leq 10$ and its exceptionally low value for $\sum_{i=1}^{n} |w_i|$. For higher dimensions, however, the number of nodes increases exponentially, resulting in a strong performance degradation.

4 Testing

Numerical tests have been performed using the test function package proposed by Genz [20]. This package defines six function families, each of them characterized by some peculiarity. Table 3 gives an overview of these functions.

Table 3. Genz's test integrand families

Integrand Family	$\|\boldsymbol{a}\|_1$	Attribute
$f_1(\boldsymbol{x}) := \cos\left(2\pi u_1 + \sum_{i=1}^{s} a_i x_i\right)$	$\frac{110}{\sqrt{s^3}}$	Oscillatory
$f_2(\boldsymbol{x}) := \prod_{i=1}^{s} \frac{1}{a_i^{-2} + (x_i - u_i)^2}$	$\frac{600}{s^2}$	Product Peak
$f_3(\boldsymbol{x}) := \left(1 + \sum_{i=1}^{s} a_i x_i\right)^{-(s+1)}$	$\frac{600}{s^2}$	Corner Peak
$f_4(\boldsymbol{x}) := \exp\left(-\sum_{i=1}^{s} a_i^2 (x_i - u_i)^2\right)$	$\frac{100}{s}$	Gaussian
$f_5(\boldsymbol{x}) := \exp\left(-\sum_{i=1}^{s} a_i \lvert x_i - u_i\rvert\right)$	$\frac{150}{s^2}$	$\mathcal{C}_0$ Function
$f_6(\boldsymbol{x}) := \begin{cases} 0 & x_1 > u_1 \vee x_2 > u_2 \\ \exp\left(\sum_{i=1}^{s} a_i x_i\right) & \text{otherwise} \end{cases}$	$\frac{100}{s^2}$	Discontinuous

For each family, $n = 20$ instances are created by choosing unaffective and affective parameters u_i and a_i pseudo-randomly from $[\frac{1}{20}, 1 - \frac{1}{20}]$. Afterwards the vector of affective parameters $\boldsymbol{a} = (a_1, \ldots, a_s)$ is scaled so that $\|\boldsymbol{a}\|_1$ meets the requested difficulty as specified in Table 3.

For each instance k ($1 \le k \le n = 20$) of an integrand family, the error e_k relative to the average magnitude of the integral of the current function family is calculated by the formula

$$e_k = \frac{|\mathrm{I}f_k - \mathrm{Q}f_k|}{\frac{1}{n}\sum_{i=1}^{n} |\mathrm{I}f_i|} \qquad \text{for } k = 1, \ldots, n \ .$$

For easier interpretation of e_k, the number of correct digits d_k in the result is obtained by the formula

$$d_k = -\log_{10} e_k \qquad \text{for } k = 1, \ldots, n \ .$$

Based on these values derived for each test integrand instance, statistical methods are used to evaluate the integrand family. The following charts show mean values with error bars based on standard deviation.

5 Results

All algorithms are implemented in a C++ program using MPI for inter-process communication. Standard double precision floating point numbers are used for all numerical calculations, forcing an upper bound of about 15 decimal digits on the accuracy of all algorithms. Tests are performed on an SGI Power Challenge GR located at the RIST++ (University of Salzburg), based on 20 R10000 MIPS processors.

We have chosen 20 test functions from each family in dimension 5, 7, 10, 15, 20, 25, 30, and 40. This set of functions is used to test all parallel algorithms, running on 2, 4, 8, and 16 processors. All calculations are also performed by the sequential algorithms to measure speed-up and evaluate the accuracy of the parallel algorithms. The number of allowed integrand evaluations is raised by a factor of 2 up to a maximal number of 2^{25} evaluations.

5.1 Which Algorithm Performs Best?

If quasi-Monte Carlo or an adaptive algorithm performs best, depends highly on the integrand and its dimension. While quasi-Monte Carlo degrades little for high dimensions and non-smooth integrands, the domain of adaptive algorithms is clearly that of smooth integrands in low dimensions. Which cubature rule leads to the best results is also dependent on the integrand and the dimension: While 9-7 works great for smooth integrands, 7-5-5 is better for discontinuous functions, while 7-5 is especially powerful in low dimensions.

Figure 1 shows which algorithm performs best for a given integrand family in a given dimension. If two algorithms are reported as "best" for a given problem, the one in the major part of the field achieved the best performance, but the one in the lower right corner is expected to beat the first one eventually, if the number of allowed integrand evaluations is increased beyond 2^{25}.

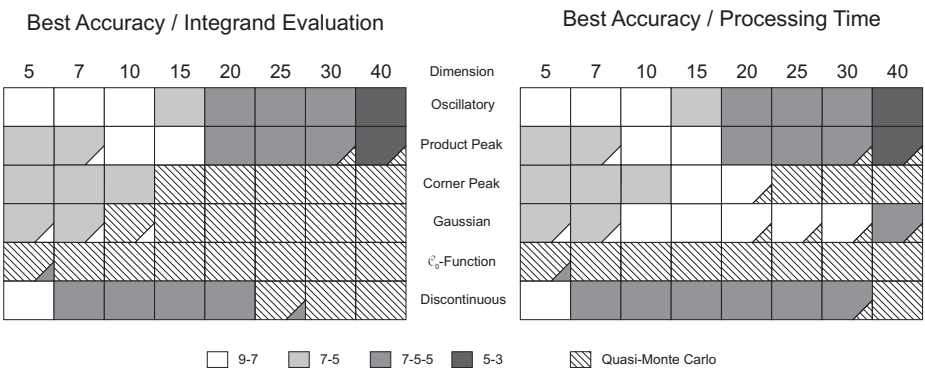


Fig. 1. Best algorithm depending on integrand family and dimension

The left chart shows which algorithm achieves the highest accuracy per integrand evaluation, while the right chart measures accuracy depending on execution time. So the right chart is most appropriate for integrands that are fast to evaluate (like the test integrands used here), while the left chart should be used for integrands that require expensive calculations. In this case the time required for integrand evaluations will dominate the total calculation time, making it most important to achieve optimal accuracy with a minimum number of integrand evaluations. The difference between these two charts is due to the different speed of the abscissa set generation for quasi-Monte Carlo and cubature rules: The time for generating a single point increases linearly with the dimension for the quasi Monte-Carlo algorithm, while the adaptive algorithm actually speeds up for increasing dimensions due to smaller region collections.

Both charts show the results for parallel algorithms running on 16 processing nodes. However, for a different number of processors, or even for the sequential algorithm, the charts are almost identical. This proves that both parallel algorithms scale equally well to at least 16 processing nodes. It follows that choosing the best algorithm does not depend on the number of processing nodes available, but only on the dimensionality of the problem and the properties of the integrand function.

5.2 Details and Discussion

This section will show in more detail how the results in Figure 1 have been obtained. For each function family and dimension, two charts have been created: The first one showing the number of correct digits depending on the number of integrand evaluation, the other one depending on execution time. Due to the huge amount of data, only a few examples can be discussed here.

The left chart in Figure 2 shows the results for Genz function f_1 (Oscillatory) for $s = 40$. This integrand is very smooth, so the adaptive algorithm performs better than quasi-Monte Carlo even for a dimension as high as $s = 40$. Only two adaptive algorithms can be seen in this chart, because the cubature rules 9-7 and 7-5 require too many points to be evaluated even a single time. 5-3 and 7-5-5, however, proof to be optimal even for high dimensions if the integrand is smooth enough.

The cubature rule showing the best performance depends on the dimension: Due to the smoothness of f_1, 9-7 performs better than 7-5 for s up to 10. For $s \geq 20$, 7-5-5 performs best and is overtaken by 5-3 for $s = 40$, because of the significantly smaller number of integrand evaluation this rule requires in this dimension.

For Genz function f_4 (Gaussian), the adaptive algorithm with the cubature rule 7-5-5 shows the best performance for dimensions up to $s = 10$. For dimensions $s = 15$ and up, however, all cubature rules are inferior to the quasi-Monte Carlo algorithm. The right chart in Figure 2 contains the result for $s = 15$, showing quasi-Monte Carlo integration superior to all four cubature rule based algorithms.

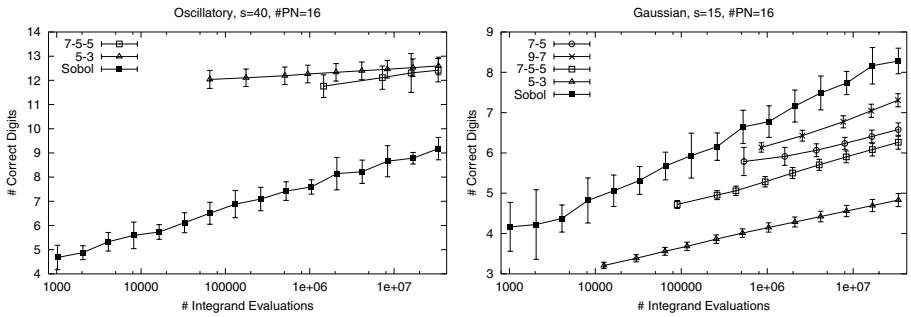


Fig. 2. Results for f_1 Oscillatory (left) and for f_4 Gaussian (right)

The left chart in Figure 3 shows Genz function f_5 ($\mathcal{C}_0$-Function) for $s = 5$. For this type of integrand, with f not differentiable on s hyperplanes, the quasi-Monte Carlo algorithm is optimal for all dimensions. For $s = 5$, it seems possible that 7-5-5 may converge faster if the number of abscissas increases beyond 2^{25}. For higher dimensions, however, the adaptive algorithms are completely outperformed.

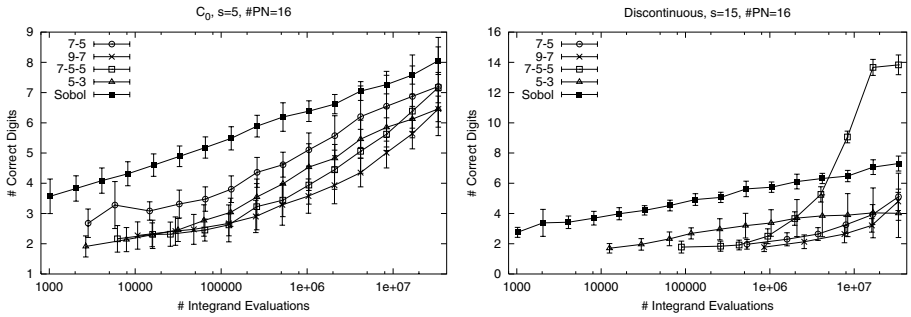


Fig. 3. Results for f_5 $\mathcal{C}_0$-Function (left) and for f_6 Discontinuous (right)

The right chart in Figure 3 shows the results for Genz function f_6 (Discontinuous) for $s = 15$. It would be reasonable to assume that the adaptive algorithms perform even worse here than for f_5. However, this is not the case. The solution to this paradox is that f_5 is discontinuous only on two hyperplanes ($x_1 = u_1 \vee x_2 = u_2$), while f_5 in not differentiable on s hyperplanes. Especially the adaptive algorithm 7-5-5 with its additional basic rule for error detection can cope with this situation, is able to refine on the regions with discontinuity and can beat the quasi-Monte Carlo algorithm even in dimensions as high as $s = 25$.

6 Conclusion

Algorithms based on quasi-Monte Carlo techniques as well as adaptive algorithms are suitable approaches for numerical integration up to at least $s = 40$ dimensions. Both algorithms can be implemented efficiently on parallel systems and provide good speedups up to at least 16 processing nodes.

If quasi-Monte Carlo or adaptive algorithms perform better depends on the dimension s of the integration problem, but also on the smoothness of the integrand. For smooth integrands, adaptive algorithms may outperform quasi-Monte Carlo techniques in dimensions as high as $s = 40$. For discontinuous or $\mathcal{C}_0$-functions, on the other hand, quasi-Monte Carlo may be superior for dimensions as low as $s = 5$.

The performance of adaptive algorithms depends highly on the cubature formula the algorithm is based on. Depending on the dimension and the type of integrand, different cubature rules should be used.

7 Acknowledgments

This work was partially supported by Österreichische Nationalbank, Jubiläumsfonds project no. 6788.

References

1. A. Krommer and C. Überhuber. *Numerical Integration on Advanced Computer Systems*. Number 848 in Lecture Notes in Computer Science. Springer-Verlag, Berlin, Heidelberg, New York, Tokyo, 1994.
2. A. Krommer and C. Überhuber. *Computationl integration*. SIAM Society for Industrial and Applied Mathematics, Philadelphia, USA, 1998.
3. I. Sobol. On the distribution of points in a cube and the approximate evaluation of integrals. *U. S. S. R. Computational Mathematics and Mathematical Physics*, 7(4):86–112, 1967.
4. I. Sobol. Uniformly distributed sequences with an additional uniform property. *U. S. S. R. Computational Mathematics and Mathematical Physics*, 16:236–242, 1976.
5. I. Antonov and V. Saleev. An economic method of computing LP_τ-sequences. *U. S. S. R. Computational Mathematics and Mathematical Physics*, 19(1):252–256, 1979.
6. V. Miller and G. Davis. Adaptive quadrature on a message-passing multiprocessor. *Journal of Parallel and Distributed Computing*, 14:417–425, 1992.
7. R. Čiegis, R. Šablinskas, and J. Waśniewski. Numerical integration on distributed-memory parallel systems. *Informatica*, 9(2):123–140, 1998.
8. R. Schürer. Adaptive numerical integration on message-passing systems. In G. Okša, R. Trobec, A. Uhl, M. Vajteršic, R. Wyrzykowski, and P. Zinterhof, editors, *Proceedings of the International Workshop Parallel Numerics ParNum 2000*, pages 93–101. Department of Scientific Computing, Salzburg University and Department of Informatics, Slovak Academy of Science, 2000.

9. A. Genz. The numerical evaluation of multiple integrals on parallel computers. In P. Keast and G. Fairweather, editors, *Numerical Integration. Recent developments, software and applications*, number C 203 in ASI Ser., pages 219–229. NATO Adv. Res. Workshop, Halifax/Canada, 1987.
10. J. Bull and T. Freeman. Parallel algorithms for multi-dimensional integration. *Parallel and Distributed Computing Practices*, 1(1):89–102, 1998.
11. M. D'Apuzzo, M. Lapegna, and A. Murli. Scalability and load balancing in adaptive algorithms for multidimensional integration. *Parallel Computing*, 23:1199–1210, 1997.
12. I. Gladwell and M. Napierala. Comparing parallel multidimensional integraion algorithms. *Parallel and Distributed Computing Practices*, 1(1):103–122, 1998.
13. A. Stroud. Remarks on the disposition of points in numerical integration formulas. *Mathematical Tables and other Aids to Computation*, 11:257–261, 1957.
14. P. Hammer and A. Stroud. Numerical evaluation of multiple integrals II. *Mathematical Tables and other Aids to Computation*, 12(64):272–280, 1958.
15. A. Stroud. Extensions of symmetric integration formulas. *Mathematics of Computation*, 22:271–274, 1968.
16. G. Phillips. Numerical integration over an n-dimensional rectangular region. *The Computer Journal*, 10:297–299, 1967.
17. F. Stenger. Numerical integration in n dimensions, 1963.
18. A. Stroud. *Approximate Calculation of Multiple Integrals.* Prentice-Hall, Englewood Cliffs, NJ, USA, 1971.
19. A. Genz and A. Malik. Remarks on algorithm 006: An adaptive algorithm for numerical integration over an n-dimensional rectangular region. *Journal of Computational and Applied Mathematics*, 6(4):295–302, 1980.
20. A. Genz. Testing multidimensional integration routines. *Tools, Methods and Languages for Scientific and Engineering Computation*, pages 81–94, 1984.

Path Integral Monte Carlo Simulations and Analytical Approximations for High-Temperature Plasmas

V. Filinov[1,2], M. Bonitz[1], D. Kremp[1], W.-D.Kraeft[3], and V. Fortov[2]

[1] Fachbereich Physik, Universität Rostock
Universitätsplatz 3, D-18051 Rostock, Germany
[2] Institute for High Energy Density, Russian Academy of Sciences,
ul. Izhorskaya 13/19, Moscow, 127412 Russia
E - mail: vs_filinov@hotmail.com
[3] Institut für Physik, Universität Greifswald
Domstr.10a, D-17487 Greifswald, Germany

Abstract. The results of analytical approximations and extensive calculations based on a path integral Monte Carlo (PIMC) scheme are presented. A new (direct) PIMC method allows for a correct determination of thermodynamic properties such as energy and equation of state of dense degenerate Coulomb systems. In this paper, we present results for dense partially ionized hydrogen at intermediate and high temperature. We give a quantitative comparison with the available results of alternative (restricted) PIMC simulations and with analytical expressions based on iterpolation formulas meeting the exact limits at low and high densities. Good agreement between the two simulations is found up to densities of the order of $10^{24}cm^{-3}$. The agreement with the analytical results is satisfactory up to densities in the range $10^{22} \dots 10^{23}cm^{-3}$.

1 Introduction

Correlated Fermi systems are of increasing interest in many fields, including plasmas, astrophysics, solids and nuclear matter, (see Kraeft et al. 1986) for an overview. Among the topics of current interest are Fermi liquids, metallic hydrogen (see DaSilva et al. 1997), plasma phase transition (see Schlanges et al. 1995), bound states etc. In such many particle quantum systems, the Coulomb interaction is essential. There has been significant progress in recent years to study these systems theoretically, and especially numerically, (see e.g. Bonitz (Ed.) 2000, Zamalin et al. 1977, Filinov, A. V. et al. 2000). A theoretical framework which is particularly well suited to describe thermodynamic properties in the region of strong coupling and degeneracy is the path integral quantum Monte Carlo (PIMC) method. There has been remarkable recent progress in applying these techniques to Fermi systems. However, these simulations are essentially hampered by the fermion sign problem. To overcome this difficulty, several strategies have been developed to simulate macroscopic Coulomb systems (see Militzer and Pollock 2000, Militzer and Ceperley 2000, and Militzer

V.N. Alexandrov et al. (Eds.): ICCS 2001, LNCS 2073, pp. 1272–1281, 2001.
© Springer-Verlag Berlin Heidelberg 2001

2000): the first is the restricted PIMC concept where additional assumptions on the density operator $\hat{\rho}$ are introduced which reduce the sum over permutations to even (positive) contributions only. This requires the knowledge of the nodes of the density matrix which is available only in a few special cases. However, for interacting macroscopic systems, these nodes are known only approximately, (see, e.g., Militzer and Pollock 2000 and Militzer and Ceperley 2000), and the accuracy of the results is difficult to assess from within this scheme.

Recently, we have published a new path integral representation for the N-particle density operator (see Filinov, V. S., et al. 2000, Bonitz (Ed.) 2000, Filinov, V. S., et al. 2001), which allows for *direct Fermionic path integral Monte Carlo* simulations of dense plasmas in a broad range of densities and temperatures. Using this concept we computed the pressure (equation of state, EOS), the energy, and the pair distribution functions of a dense partially ionized and dissociated electron–proten plasma (see Filinov, V. S., et al. 2000). In this region no reliable data are available from other theories such as density functional theory or quantum statistics (see, e.g., Kraeft et al. 1986), which would allow for an unambiguous test. Therefore, it is of high interest to perform quantitative comparisons of analytical results and independent numerical simulations, such as restricted and direct fermionic PIMC, which is the aim of this paper.

2 Path Integral Representation of Thermodynamic Quantities

We now briefly outline the idea of our direct PIMC scheme. All thermodynamic properties of a two-component plasma are defined by the partition function Z which, for the case of N_e electrons and N_p protons, is given by

$$Z(N_e, N_p, V, \beta) = \frac{Q(N_e, N_p, \beta)}{N_e! N_p!},$$

$$\text{with} \qquad Q(N_e, N_p, \beta) = \sum_{\sigma} \int_V dq\, dr\, \rho(q, r, \sigma; \beta), \tag{1}$$

where $\beta = 1/k_B T$. The exact density matrix is, for a quantum system, in general, not known but can be constructed using a path integral representation (see Feynman and Hibbs 1965),

$$\int_V dR^{(0)} \sum_{\sigma} \rho(R^{(0)}, \sigma; \beta) = \int_V dR^{(0)} \dots dR^{(n)}\, \rho^{(1)} \cdot \rho^{(2)} \dots \rho^{(n)}$$

$$\times \sum_{\sigma} \sum_{P} (\pm 1)^{\kappa_P}\, \mathcal{S}(\sigma, \hat{P}\sigma')\, \hat{P} \rho^{(n+1)}, \tag{2}$$

where $\rho^{(i)} \equiv \rho\left(R^{(i-1)}, R^{(i)}; \Delta\beta\right) \equiv \langle R^{(i-1)} | e^{-\Delta\beta \hat{H}} | R^{(i)} \rangle$, whereas $\Delta\beta \equiv \beta/(n+1)$. $\hat{H}$ is the Hamilton operator, $\hat{H} = \hat{K} + \hat{U}_c$, containing kinetic and potential energy contributions with $\hat{U}_c = \hat{U}_c^p + \hat{U}_c^e + \hat{U}_c^{ep}$ being the sum of the Coulomb potentials between protons (p), electrons (e) and electrons and protons (ep). Further,

$R^{(i)} = (q^{(i)}, r^{(i)}) \equiv (R_p^{(i)}, R_e^{(i)})$, for $i = 1, \ldots n+1$, $R^{(0)} \equiv (q, r) \equiv (R_p^{(0)}, R_e^{(0)})$. Also, $R^{(n+1)} \equiv R^{(0)}$ and $\sigma' = \sigma$, i.e., the particles are represented by closed Fermionic loops with the coordinates (beads) $[R] \equiv [R^{(0)}; R^{(1)}; \ldots; R^{(n)}; R^{(n+1)}]$, where r and q denote the electron and proton coordinates, respectively. The spin gives rise to the spin part of the density matrix $\mathcal{S}$, whereas exchange effects are accounted for by the permutation operator $\hat{P}$, which acts on the electron coordinates and spin, and the sum over the permutations with parity κ_P. In the fermionic case (minus sign), the sum contains $N_e!/2$ positive and negative terms leading to the notorious sign problem. Due to the large mass difference of electrons and protons, the exchange of the latter is not included.

Recently, we have derived a new representation for the high–temperature density matrices $\rho^{(i)}$ in eq.(2) (see Filinov, V. S., et al. 2000) which is well suited for direct PIMC simulations. A crucial point is that the electron–proton interaction can be described by an (effective) quantum pair potential Φ^{ep} (*Kelbg–potential*, see Kelbg 1964). For details of the derivation see Filinov, V. S., et al. 2000. Here, we present only the final result for the energy and for the EOS. Consider first the energy:

$$\beta E = \frac{3}{2}(N_e + N_p) + \frac{1}{Q}\frac{1}{\lambda_p^{3N_p}\Delta\lambda_e^{3N_e}} \sum_{s=0}^{N_e} \int dq\, dr\, d\xi\, \rho_s(q,[r],\beta) \times$$
$$\left\{ \sum_{p<t}^{N_p} \frac{\beta e^2}{|q_{pt}|} + \sum_{l=0}^{n} \left[\sum_{p<t}^{N_e} \frac{\Delta\beta e^2}{|r_{pt}^l|} + \sum_{p=1}^{N_p}\sum_{t=1}^{N_e} \Psi_l^{ep} \right] \right.$$
$$+ \sum_{l=1}^{n} \left[-\sum_{p<t}^{N_e} C_{pt}^l \frac{\Delta\beta e^2}{|r_{pt}^l|^2} + \sum_{p=1}^{N_p}\sum_{t=1}^{N_e} D_{pt}^l \frac{\partial \Delta\beta\Phi^{ep}}{\partial |x_{pt}^l|} \right]$$
$$\left. - \frac{1}{\det|\psi_{ab}^{n,1}|_s} \frac{\partial \det|\psi_{ab}^{n,1}|_s}{\partial\beta} \right\},$$
$$\text{with} \quad C_{pt}^l = \frac{\langle r_{pt}^l | y_{pt}^l \rangle}{2|r_{pt}^l|}, \qquad D_{pt}^l = \frac{\langle x_{pt}^l | y_p^l \rangle}{2|x_{pt}^l|}, \tag{3}$$

and $\Psi_l^{ep} \equiv \Delta\beta\partial[\beta'\Phi^{ep}(|x_{pt}^l|,\beta')]/\partial\beta'|_{\beta'=\Delta\beta}$ contains the electron-proton Kelbg potential Φ^{ep}. Here, $\langle \ldots | \ldots \rangle$ denotes the scalar product, and q_{pt}, r_{pt} and x_{pt} are differences of two coordinate vectors: $q_{pt} \equiv q_p - q_t$, $r_{pt} \equiv r_p - r_t$, $x_{pt} \equiv r_p - q_t$, $r_{pt}^l = r_{pt} + y_{pt}^l$, $x_{pt}^l \equiv x_{pt} + y_p^l$ and $y_{pt}^l \equiv y_p^l - y_t^l$, with $y_a^n = \Delta\lambda_e \sum_{k=1}^{n} \xi_a^{(k)}$ and $\Delta\lambda_a^2 = 2\pi\hbar^2\Delta\beta/m_a$. We introduced dimensionless distances between neighboring vertices on the loop, $\xi^{(1)}, \ldots \xi^{(n)}$, thus, explicitly, $[r] \equiv [r; y_e^{(1)}; y_e^{(2)}; \ldots]$.

The density matrix ρ_s is given by

$$\rho_s(q,[r],\beta) = C_{N_e}^s\, e^{-\beta U(q,[r],\beta)} \prod_{l=1}^{n}\prod_{p=1}^{N_e} \phi_{pp}^l \det|\psi_{ab}^{n,1}|_s, \tag{4}$$

where $U(q,[r],\beta) = U_c^p(q) + \{U^e([r],\Delta\beta) + U^{ep}(q,[r],\Delta\beta)\}/(n+1)$ and $\phi_{pp}^l \equiv \exp[-\pi|\xi_p^{(l)}|^2]$. We underline that the density matrix (4) does not contain an

explicit sum over the permutations and thus no sum of terms with alternating sign. Instead, the whole exchange problem is contained in a single exchange matrix given by

$$||\psi_{ab}^{n,1}||_s \equiv ||e^{-\frac{\pi}{\Delta\lambda_e^2}|(r_a-r_b)+y_a^n|^2}||_s. \tag{5}$$

As a result of the spin summation, the matrix carries a subscript s denoting the number of electrons having the same spin projection. For more detail, (see Filinov, V. S., et al. 2000, Bonitz (Ed.) 2000). In a similar way, we obtain the result for the equation of state,

$$\begin{aligned}\frac{\beta pV}{N_e+N_p} = 1 + \frac{1}{N_e+N_p}\frac{(3Q)^{-1}}{\lambda_p^{3N_p}\Delta\lambda_e^{3N_e}}\sum_{s=0}^{N_e}\int dq\,dr\,d\xi\,\rho_s(q,[r],\beta)\times \\ \left\{\sum_{p<t}^{N_p}\frac{\beta e^2}{|q_{pt}|}+\sum_{p<t}^{N_e}\frac{\Delta\beta e^2}{|r_{pt}|}-\sum_{p=1}^{N_p}\sum_{t=1}^{N_e}|x_{pt}|\frac{\partial\Delta\beta\Phi^{ep}}{\partial|x_{pt}|}\right. \\ +\sum_{l=1}^{n}\left[\sum_{p<t}^{N_e}A_{pt}^l\frac{\Delta\beta e^2}{|r_{pt}^l|^2}-\sum_{p=1}^{N_p}\sum_{t=1}^{N_e}B_{pt}^l\frac{\partial\Delta\beta\Phi^{ep}}{\partial|x_{pt}^l|}\right] \\ \left.+\frac{\alpha}{\det|\psi_{ab}^{n,1}|_s}\frac{\partial\det|\psi_{ab}^{n,1}|_s}{\partial\alpha}\right\}, \\ \text{with}\quad A_{pt}^l=\frac{\langle r_{pt}^l|r_{pt}\rangle}{|r_{pt}^l|},\qquad B_{pt}^l=\frac{\langle x_{pt}^l|x_{pt}\rangle}{|x_{pt}^l|}.\end{aligned} \tag{6}$$

3 Analytical Approximations for the Thermodynamic Functions of Dense Plasmas

To describe dense plasmas, it is necessary to have thermodynamic functions valid at arbitrary degeneracy. Here, we restrict ourselves to the Hartree–Fock (HF) and the Montroll–Ward (MW) contributions. This approximation is appropriate at temperatures high enough such that the Coulomb interaction is weak and the possibility of the formation of bound states is excluded. HF and MW contributions have been computed numerically (see Kraeft et al. 1986). The analytical evaluation of the MW contribution is possible in limiting situations only, namely in the low and very high density cases. In the intermediate region Padé formulae can be used to interpolate between the limiting cases. In between, the formulae are fitted to numerical data; (see Ebeling et al. 1981 , Haronska et al. 1987 and Ebeling and Richert 1985). We give the excess free energy and the interaction part of the chemical potential of the electron gas,

$$f^P = \frac{f_D - \frac{1}{4}(\pi\beta)^{-1/2}\bar{n} + 8\bar{n}^2 f_{GB}}{1+8ln\left[1+\frac{3}{64\sqrt{2}}(\pi\beta)^{1/4}\bar{n}^{1/2}\right]+8\bar{n}^2}, \tag{7}$$

and

$$\mu^P = \frac{\mu_D - \frac{1}{2}(\pi\beta)^{-1/2}\bar{n} + 8\bar{n}^2\mu_{GB}}{1 + 8ln\left[1 + \frac{1}{16\sqrt{2}}(\pi\beta)^{1/4}\bar{n}^{1/2}\right] + 8\bar{n}^2} \,. \tag{8}$$

In (7,8) Heaviside units $\hbar = \frac{e^2}{2} = 2m_e = 1$ and the dimensionless density $\bar{n} = n\Lambda^3$ were used.

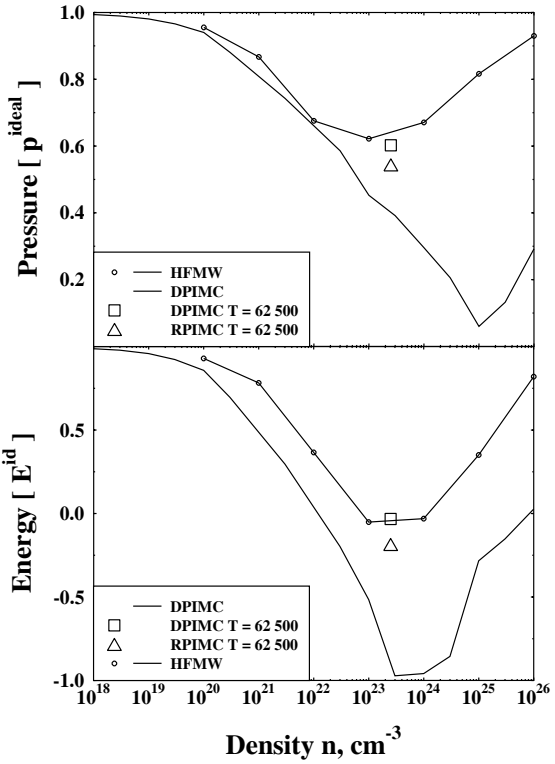


Fig. 1. Energy and pressure isotherms for $50,000K$ (solid line). Solid line with circles– Hartree Fock (HF) and Montroll–Ward (MW) approximation. Reference data: triangle – RPIMC (see Militzer et al.), square – DPIMC.

In formulae (7,8), the correct low density behaviour (Debye limiting law) is guaranteed by choosing $f_D = -(2/3)(\pi\beta)^{-1/4}\bar{n}^{1/2}$ and $\mu_D = -(\pi\beta)^{-1/4}\bar{n}^{1/2}$. The correct high degeneracy limit is recovered by using the (slightly modified) *Gell-Mann Brueckner* approximations (including Hartree–Fock) for the free energy and for the chemical potential

$$f_{GB} = -\frac{0.9163}{r_s} - 0.08883\ln\left[1 + \frac{4.9262}{r_s^{0.7}}\right] \approx -\frac{0.9163}{r_s} + 0.0622\ln r_s \,, \tag{9}$$

$$\mu_{GB} = -\frac{1.2217}{r_s} - 0.08883\ln\left[1 + \frac{6.2208}{r_s^{0.7}}\right] \approx -\frac{1.2217}{r_s} + 0.0622\ln r_s\,. \qquad (10)$$

The free energy is now equal to the *internal energy at* $T = 0$ and reads, according to Carr and Maradudin,

$$\frac{U}{N} = \frac{2.21}{r_s^2} - \frac{0.916}{r_s} + 0.0622\ln r_s - 0.096 + 0.018 r_s \ln r_s + \cdots\,.$$

The Brueckner parameter r_s is given by $r_s^3 = 3/(4\pi n)$. While the Hartree Fock term, i.e., the $1/r_s$ term in (9,10), was retained unaffected, the additional terms in these equations and in formulae (7,8) were modified, or *fitted*, respectively, such that (7,8) meet the numerical data *in between*, where the analytical limiting formulae are not applicable.

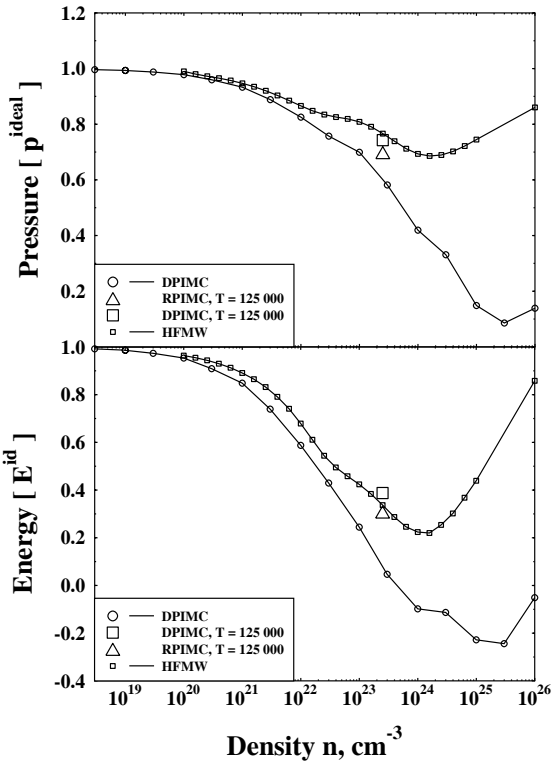


Fig. 2. Energy and pressure isotherms for $100,000K$ (solid line with circles). Solid line with small squares – Hartree Fock (HF) and Montroll–Ward (MW) approximation. Reference data: triangle – RPIMC (see Militzer et al. 2000), large square – DPIMC.

Consider now the proton contributions. In the low density regime, the proton formulae are practically the same as for the electrons, whereas in the high density

limit we adjust the formulae to (classical) Monte–Carlo (MC) data. We have for the free energy density and for the chemical potential

$$-\frac{f_p}{k_BTn_p}=\frac{(-f_p^{\rm int}/k_BTn_p)_D[1-a\tilde{n}_p^{2/3}(f_p^{\rm int}/k_BTn_p)_{MC}]}{1-a\tilde{n}_p^{1/2}[\tilde{n}_p^{1/2}/(\frac{f_p^{\rm int}}{k_BTn_P})_D+\tilde{n}_p^{1/6}(\frac{f_p^{\rm int}}{k_BTn_p})_D]} \tag{11}$$

$$-\frac{\mu_p}{k_BT}=\frac{(-\mu_p^{\rm int}/k_BT)_D[1-2a\tilde{n}_p^{2/3}(\mu_p^{\rm int}/k_BTn_p)_{MC}]}{1-2a\tilde{n}_p^{1/2}[\tilde{n}_p^{1/2}/(\frac{\mu_p^{\rm int}}{k_BT})_D+\tilde{n}_p^{1/6}(\frac{\mu_p^{\rm int}}{k_BT})_D]}\,. \tag{12}$$

We used the abbreviation a which depends only on temperature

$$a=\sqrt{\pi^3k_BT}\Big\{\frac{1}{2}\Big[1+\sqrt{\frac{k_BT}{4\pi}}\exp\Big(\frac{\sqrt{\pi}/2}{\ln(4/k_BT)^{1/6}-2\sqrt{k_BT}}\Big)\Big]-0.29931\Big\}\,. \tag{13}$$

For the protons, we introduce the dimensionless density to be used in the plasma parameter Γ, namely $\tilde{n}_p=\frac{8}{(k_BT)^3}n_p$, $\Gamma=\left(\frac{4}{3}\pi\tilde{n}_p\right)^{1/3}$. The Debye approximations (i.e., the low density case) for free energy density and chemical potential read $(-f_p^{\rm int}/k_BTn_p)_D=2.1605\tilde{n}_p^{1/2}$ and $(-\mu_p^{\rm int}/k_BT)_D=\frac{3}{2}2.1605\tilde{n}_p^{1/2}$. In the high density region, we use a fit to (classical OCP) Monte–Carlo data.

For the free energy we write

$$\begin{aligned}&(f_p^{\rm int}/k_BTn_p)_{MC}=-0.8946\Gamma+3.266\Gamma^{1/4}-0.5012\ln\Gamma-2.809\\&-\frac{r_s\tilde{n}_p^{1/3}}{1+r_s^2}\Big[0.0933+1.0941\tilde{n}_p^{-1/4}-0.343\tilde{n}_p^{-1/3}\Big]\,,\end{aligned} \tag{14}$$

and for the chemical potential

$$\begin{aligned}&(\mu_p^{\rm int}/k_BT)_{MC}=-1.1928\Gamma+3.5382\Gamma^{1/4}-0.5012\ln\Gamma-2.9761\\&-\frac{r_s\tilde{n}_p^{1/3}}{1+r_s^2}\Big[0.0933+0.8206\tilde{n}_p^{-1/4}-0.2287\tilde{n}_p^{-1/3}\Big]\,.\end{aligned} \tag{15}$$

The *correlation part* of the pressure for an H–plasma is then given by

$$p^{\rm corr}=p_e^{\rm corr}+p_p^{\rm corr}=n_e\mu_e+n_p\mu_p-f_e-f_p\,. \tag{16}$$

The contributions are determined by (7,8) and (11,12). The ideal pressure is given by Fermi integrals $I_\nu(\alpha)$

$$p^{\rm id}=k_BT\sum_a\frac{2s_a+1}{\Lambda_a^3}I_{3/2}(\alpha_a)\ ,\ \ \alpha_a=\mu_a/(k_BT)\,.$$

The internal energy may be constructed from the excess free energy $f=\frac{F}{V}$ given above in addition to the ideal part according to $U=F-T\frac{\partial F}{\partial T}\big|_{V=\rm const}$, where the ideal free energy is given by

$$F^{\rm id}=k_BTV\sum_a\frac{2s_a+1}{\Lambda_a^3}\left\{\alpha_aI_{1/2}(\alpha_a)-I_{3/2}(\alpha_a)\right\}\,.$$

At very high degeneracy, the free energy is equal to the internal energy.

4 Hydrogen Isotherms

In this section we present results for the thermodynamic functions of dense hydrogen versus density at constant temperature. The PIMC simulations have been performed as explained in Filinov, V. S., et al. 2000, and Filinov V. S., et al. 2001. Figures 1-3 show the simulation results together with the Padé results for three hydrogen isotherms $T = 50,000$; $100,000$; and $125,000K$. In all figures the agreement between numerical and analytical data is good for temperatures and densities, where the coupling parameter Γ is smaller than or equal to unity. Reference points related to RPIMC and DPIMC calculations in Fig. 1 correspond to data available for the temperature $T = 62,500K$.

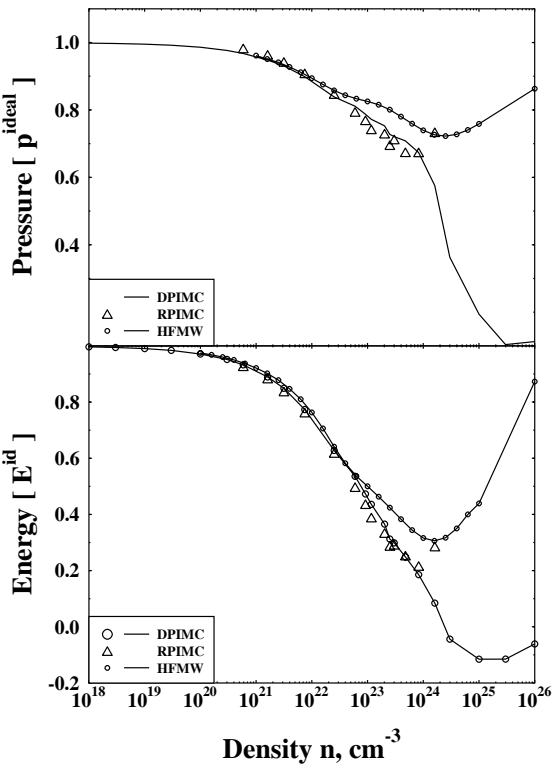


Fig. 3. Energy and pressure isotherms for $125,000K$ (lower curves, solid line with larger or without circles). Solid line with smaller circles – Hartree Fock (HF) and Montroll–Ward (MW) approximation for $125000K$. Reference data: triangles – RPIMC (see Militzer et al. 2000).

At low densities, pressure and energy are close to those of an ideal plasma. Increasing the density above $10^{19}cm^{-3}$, Coulomb interaction becomes important leading to a decrease of pressure and energy. Differences between analytical

and numerical calculations in Fig. 1 are observed for densities above $10^{22}\ cm^{-3}$ where the coupling parameter Γ exceeds unity. At temperatures of $100,000$ and $125,000K$, differences are observed for n above $5\times10^{22}cm^{-3}$. The degeneracy parameter $n_e\lambda^3$ reaches here values of 0.4. At higher densities (around $10^{24}cm^{-3}$) the degeneracy $n_e\lambda^3$ becomes larger than unity, and the interaction parts of pressure and energy decrease as compared to the respective ideal contributions, which leads to an increase of pressure and energy. At lower temperatures, this tendency is accompanied by the vanishing of bound states, i.e., a transition from a partially ionized plasma to a metal–like state. This tendency is correctly reproduced by all methods, however the density values of this increase vary. In Fig. 3 we compare our results with data from RPIMC simulations (see Militzer et al 2000). Obviously, the agreement is very good up to densities below $10^{24}cm^{-3}$.

5 Discussion

This work is devoted to a Quantum Monte Carlo study of a correlated proton-electron system with degenerate electrons. We compared our direct PIMC simulations with independent restricted PIMC results of Militzer and Ceperley and analytical formulae for isotherms corresponding to $T = 50,000$; $100,000$; and $125,000K$. The values of Γ and $n_e\Lambda_e^3$ are varying in a wide range of values. This region is of particular interest as here pressure and temperature ionization occur and, therefore, an accurate and consistent treatment of scattering and bound states is crucial. We found that the results agree sufficiently well for coupling parameters smaller or equal to unity. This is remarkable because analytical formulae, the DPIMC and RPIMC simulations are completely independent and use essentially different approximations. We, therefore, expect that these results for hydrogen are reliable which is the main result of the present paper. We hope that our simulation results allow us to derive and test improved analytical approximations in the future.

6 Acknowledgements

We acknowledge support by the Deutsche Forschungsgemeinschaft (Sonderforschungsbereich 198: M.B., D.K., and W.D.K; Mercator-Programm: V.S.F.). Our thanks are due to W. Ebeling and M. Schlanges for stimulating discussions.

References

Kraeft, W.D., Kremp, D., Ebeling, W., Röpke, G.: Quantum Statistics of Charged Particle Systems, Akademie-Verlag Berlin and Plenum New York 1986

Da Silva, I.B. et al.,: Phys. Rev. Lett. **78** (1997) 783

Schlanges, M., Bonitz, M., Tschtschjan, A.: Contrib. Plasma Phys. **35** (1995)109

Bonitz, M., (Ed.): Progress in Nonequlibrium Green's functions, World Scientific, Singapore 2000

Zamalin, V.M., Norman, G. E., Filinov, V. S.: The Monte Carlo Method in Statistical Thermodynamics, Nauka, Moscow 1977 (in Russian).
Filinov, A. V., Lozovik, Yu. E., Bonitz, M.: phys. stat. sol. (b) **221** (2000) 231; Filinov, A. V., Bonitz, M., Lozovik, Yu. E.: Phys. Rev. Lett. (2001), arXiv:cond-mat/0012265
Militzer, B., Pollock, E.: Phys. Rev. E **61** (2000) 3470
Militzer, B., Ceperley, D.: Phys. Rev. Lett. **85** (2000) 1890
Militzer, B., PhD-Thesis, University of Illinois (2000)
Filinov, V. S., Bonitz, M., Fortov, V. E.: JETP Letters, **72** (2000) 245; Filinov, V. S., Fortov, V. E., Bonitz, M., Kremp, D.: Phys. Lett. A **274** (2000) 228
Filinov, V. S., Bonitz, M., Ebeling, W., Fortov, V. E.: Plasma Phys. Contr. Fusion (2001)
Feynman, R. P., Hibbs, A. R.: Quantum mechanics and path integrals, 1965, McGraw-Hill, New York.
Kelbg, G.: Ann. Physik, **12** (1963) 219; **13** (1964) 354; **14** (1964) 394
Ebeling, W., Richert, W., Kraeft, W. D., Stolzmann, W.: phys.stat.sol.(b) **104** (1981) 193
Haronska, P., Kremp, D., Schlanges, M: Wiss. Z. Univ. Rostock, MN Reihe **36** (1987) 98
Ebeling, W., Richert, W.: Phys. Lett. A **108** (1985) 80; phys. stat sol. (b) **128** (1985) 67

A Feynman-Kac Path-Integral Implementation for Poisson's Equation

Chi-Ok Hwang and Michael Mascagni

Department of Computer Science, Florida State University, 203 Love Building
Tallahassee, FL 32306-4530

Abstract. This study presents a Feynman-Kac path-integral implementation for solving the Dirichlet problem for Poisson's equation. The algorithm is a modified "walk on spheres" (WOS) that includes the Feynman-Kac path-integral contribution for the source term. In our approach, we use the Poisson kernel instead of simulating Brownian trajectories in detail to implement the path-integral computation. We derive this approach and provide results from a numerical experiment on a two-dimensional problem as verification of the method.

1 Introduction

Since Müller proposed the "walk on spheres" (WOS) method for solving the Dirichlet boundary value problems for the Laplace equation [1], WOS has been a popular method. In addition, this random-walk based approach has been extended to solve other, more complicated, partial differential equations including Poisson's equation, and the linearized Poisson-Boltzmann equation [2–8]. In WOS, instead of using detailed Brownian trajectories inside the domain, discrete jumps are made using the uniform first-passage probability distribution of the sphere. In this paper, this WOS method is combined with the Feynman-Kac formulation to solve the Dirichlet boundary value problem for Poisson's equation. Even though the Feynman-Kac method is well known among mathematicians and mathematical physicists, [9–11], as a computational technique, it has not been much implemented, even for simple cases, despite the fact that some modified WOS methods are mathematically derivable from the Feynman-Kac formulation [4]. We thus feel that it is worthwhile to implement the Feynman-Kac formulation for some simple problems to show its utility. Here, we implement the Feynman-Kac formulation for a simple Poisson problem. Instead of simulating the detailed Brownian trajectory, we use discrete WOS jumps together with the Poisson kernel to incorporate the source term in the Poisson equation. In previous work of others, different Green's functions have been used [2, 4, 8] for the source term in the Poisson's equation, but the Poisson kernel has never been used. Here, we interpret the Poisson kernel as the probability density distribution of a Brownian trajectory inside a ball during its passage from the ball's center to an exit point on the ball's boundary. This is used in place of a direct simulation of the Brownian trajectory because the Poisson kernel gives the Brownian walker's density distribution inside the ball.

V.N. Alexandrov et al. (Eds.): ICCS 2001, LNCS 2073, pp. 1282–1288, 2001.
© Springer-Verlag Berlin Heidelberg 2001

This paper is organized as follows. In Section 2, we explain how to implement the Feynman-Kac path-integral method for the Dirichlet problem for Poisson's equation. In Section 3, we give a numerical example. In Section 4, conclusions are presented and future work is discussed.

2 Modified "walk on spheres"

In this section, we explain how to combine the WOS method [1] with the Feynman-Kac path-integral representation for solving the Dirichlet problem for Poisson's equation. Our implementation is based on the well-known Feynman-Kac representation of the solution to the Dirichlet problem for Poisson's equation. Recall that the Dirichlet problem for the Poisson's equation is:

$$\frac{1}{2}\Delta u(\mathbf{x}) = -q(\mathbf{x}), \qquad \mathbf{x} \in \Omega \tag{1}$$

$$u(\mathbf{x}) = f(\mathbf{x}), \qquad \mathbf{x} \in \partial\Omega. \tag{2}$$

The solution to this problem, given in the form of the path-integral with respect to standard Brownian motion $X_t^{\mathbf{x}}$, is as follows [9, 10]:

$$u(\mathbf{x}) = E[\int_0^{\tau_D^{\mathbf{x}}} q(X_t^{\mathbf{x}})dt] + E[f(X_{\tau_D^{\mathbf{x}}}^{\mathbf{x}})], \tag{3}$$

where $\tau_D^{\mathbf{x}} = \{t : X_t^{\mathbf{x}} \in \partial\Omega\}$ is the first passage time and $X_{\tau_D^{\mathbf{x}}}^{\mathbf{x}}$ is the first passage location on the boundary, $\partial\Omega$[1].

Instead of simulating the detailed irregular motion of the Brownian trajectories, we use the Poisson kernel for a ball [10] with WOS as a probability density function:

$$K(\mathbf{x}, \mathbf{z}) = \frac{\Gamma(d/2)}{2\pi^{d/2}r} \frac{r^2 - |\mathbf{x} - \mathbf{a}|^2}{|\mathbf{x} - \mathbf{z}|^d}, \qquad \mathbf{x} \in B(\mathbf{a}, r), \quad \mathbf{z} \in S(\mathbf{a}, r). \tag{4}$$

Here, $B(\mathbf{a}, r)$ is a ball with center $\mathbf{a}$ and radius r, $\mathbf{z}$ is the first passage location on the surface of the ball, $S(\mathbf{a}, r)$, and d is dimension. We interpret the Poisson kernel as the probability density distribution of the Brownian trajectory inside the ball during its passage from the center to the boundary.

We construct a Brownian trajectory as a sequence of discrete jumps from ball centers to ball surfaces. Using the Poisson kernel, the first term of Eq. 3 for each ith ball of a WOS (Brownian) trajectory becomes

$$E[E\{\tau_D^{\mathbf{x_i}}\} \int_{B_i} q(\mathbf{x})K(\mathbf{x}, \mathbf{z})d\mathbf{x}]. \tag{5}$$

[1] Here, we assume that $E[\tau_D^{\mathbf{x}}] < \infty$ for all $\mathbf{x} \in \Omega$, $f(\mathbf{x})$ and $q(\mathbf{x})$ are continuous and bounded, and that the boundary, $\partial\Omega$, is sufficiently smooth so as to ensure the existence of a unique solution, $u(\mathbf{x})$, that has bounded, continuous, first-order and second-order partial derivatives in any interior subdomain [9, 10].

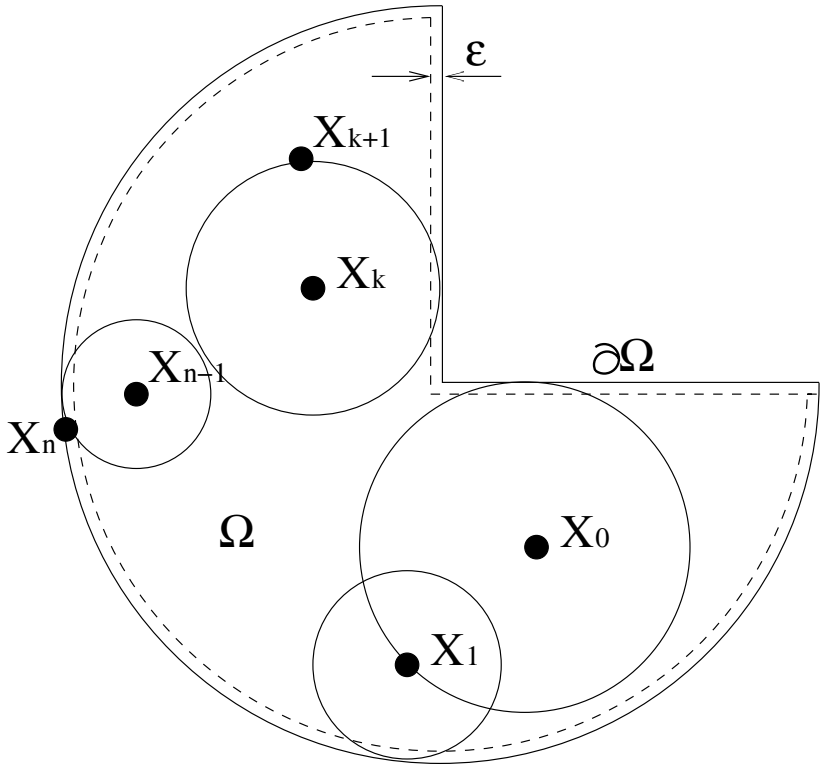


Fig. 1. Modified WOS. X_0, X_1, ..., X_k, ..., X_n are a series of discrete jumps of a Brownian trajectory which terminates on absorption in the ϵ-absorption layer.

Here, B_i is the volume of the ith ball and so the proceeding is a volume integral. It is also well known that the mean first passage time in this situation is $E[\tau_D^{\mathbf{x}_D^i}] = r_i^2/2d$ in d dimensions [12, 13]. Notice that the integral is improper at $\mathbf{z}$ when $\mathbf{x} = \mathbf{z}$, the exit point.

Eq. 5 readily permits the use of WOS to eliminate the need to compute the detailed Brownian trajectory. Instead, a series of discrete jumps in continuous space terminating on the boundary, $\partial\Omega$, is used. Jumping from ball to ball never permits a trajectory to land exactly on the boundary. Thus we use the standard WOS approach of "fattening" the boundary by ϵ to create a capture region that is used to terminate the walk [2]. The error associated with this approximation has been theoretically estimated in previous WOS methods [6, 8].

We wish to compute the solution to the Dirichlet problem for Poisson's equation at $\mathbf{x_0}$. For each Brownian trajectory starting at $\mathbf{x_0}$, with an ϵ-absorption layer, we accumulate the internal contribution for each ball and the functional value of the boundary condition at the final exit location on $\partial\Omega$. And so, an estimate for the solution at $\mathbf{x_0}$ is given by the statistic

Table 1.

r	θ	Exact	Monte Carlo	variance	Average number of steps
0.1244	-0.7906	0.8623	0.8676	0.0971	13.04
0.2320	-0.0274	0.9678	0.9694	0.0039	8.70
0.2187	-3.3975	0.4308	0.4348	0.1826	13.54
0.1476	-4.1617	0.4695	0.4564	0.0886	13.01
0.0129	-1.4790	0.8890	0.8843	0.0654	12.24

$$S_N = \frac{1}{N}\sum_{i=1}^{N} Z_i, \tag{6}$$

where N is the number of trajectories and each statistic, Z_i is given by

$$Z_i = \sum_{i=1}^{n_i}[E\{\tau_D^{\mathbf{x_i}}\}\int_{B(\mathbf{x_i},r_i)} q(\mathbf{x})K(\mathbf{x},\mathbf{z})d\mathbf{x} + f(X_{\tau_D}^{\mathbf{x}})]. \tag{7}$$

Here, n_i is the number of WOS steps needed for the ith Brownian trajectory to terminate in the ϵ-absorption layer.

3 Numerical Experiments

In this section, we demonstrate our Feynman-Kac implementation by solving numerically a boundary value problem for Poisson's equation. We use as our domain, Ω, the unit disk minus the first quadrant, which was used in previous research by DeLaurentis and Romero [8] (See Fig 1.):

$$\Omega = \{(r,\theta) : 0 < r < 1, -3\pi/2 < \theta < 0\}. \tag{8}$$

We consider the Poisson equation:

$$\frac{1}{2}\Delta u(\mathbf{x}) = -(1 - \frac{r^2}{2})e^{-r^2/2} \tag{9}$$

with the boundary conditions $u(r,0) = e(-r^2/2)$, $u(r,-3\pi/2) = -r^{1/3} + e(-r^2/2)$, and $u(1,\theta) = \sin(\theta/3) + e^{-1/2}$. The known analytic solution is

$$u(r,\theta) = r^{1/3}\sin(\theta/3) + e^{-r^2/2}. \tag{10}$$

For this two-dimensional problem, the Poisson kernel is

$$K(\mathbf{x},\mathbf{z}) = \frac{1}{2\pi r}\frac{r^2 - |\mathbf{x}-\mathbf{a}|^2}{|\mathbf{x}-\mathbf{z}|^2}, \qquad \mathbf{x} \in B(\mathbf{a},r), \quad \mathbf{z} \in S(\mathbf{a},r), \tag{11}$$

and one random estimate is

$$Z_i = \sum_{i=1}^{n} \frac{r_i^2}{4}\int_{B(\mathbf{x_i},r_i)} q(\mathbf{x})K(\mathbf{x},\mathbf{z})d\mathbf{x} + f(X_{\tau_D}^{\mathbf{x}}). \tag{12}$$

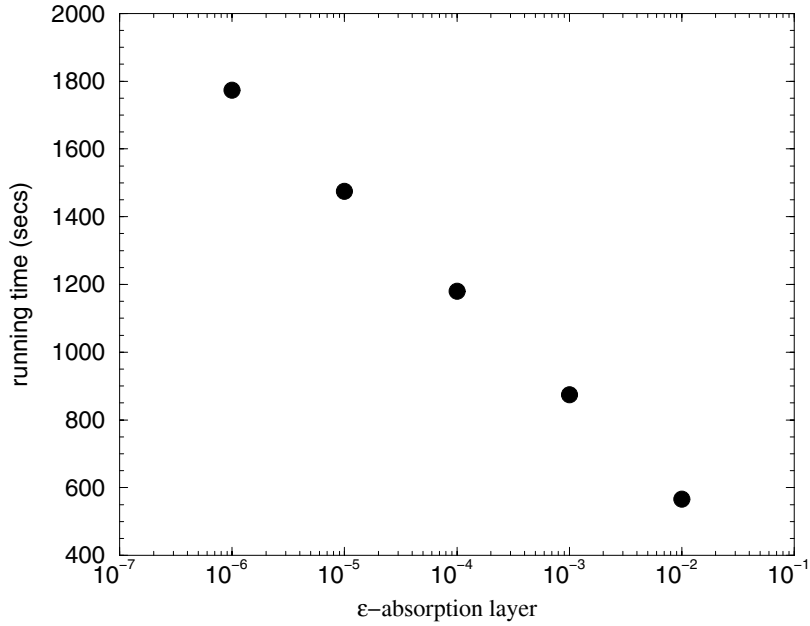


Fig. 2. Running time *v.s.* the thickness of the ϵ-absorption layer. This shows the usual relation for WOS, running time (proportional to number of WOS steps for each Brownian trajectory) on the order of $|\log \epsilon|$.

Table I shows our simulation results for the solution at five different points. The absorption layer thickness is $\epsilon = 10^{-4}$, and the number of trajectories for each run is $N = 10^3$.

The errors associated with this implementation are (1) the error associated with the number of trajectories (sampling error), (2) the error associated with the ϵ-absorption layer, and (3) the error associated with the integration used for the source term. We can reduce the statistical sampling error by increasing the number of trajectories. The error associated with the ϵ-absorption layer can be reduced by reducing ϵ, the ϵ-absorption layer thickness. However, increasing the number of trajectories will increase the running time linearly, while reducing ϵ will increase running time on the order of $|\log \epsilon|$. In Fig. 2, we see the usual relationship between running time and the thickness of the ϵ-absorption layer in WOS: on the order of $|\log \epsilon|$. The reason for this is that the running time increases proportionally to the number of WOS steps, n_i, as an integration is required for the source term in each WOS step. The error from the ϵ-absorption layer can be investigated empirically if we have enough trajectories so that the statistical sampling error is much smaller than the error from the ϵ-absorption layer. Fig. 3 shows the empirical results with 10^6 Brownian trajectories: the ϵ-layer error grows linearly in ϵ for small ϵ.

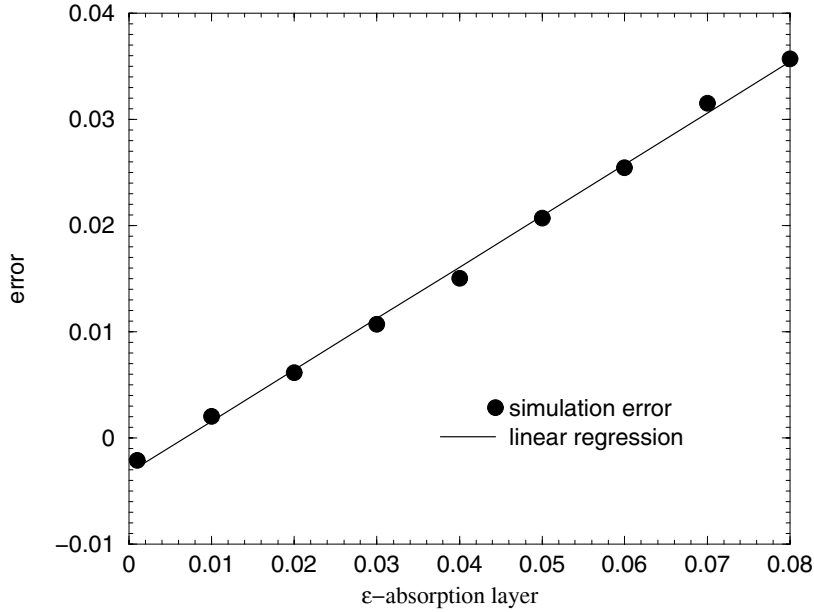


Fig. 3. Error arising from the ϵ-absorption layer with 10^6 Brownian trajectories. The error is linear in ϵ.

4 Conclusions and Future Work

In this study, we implemented the Feynman-Kac path-integral representation of the solution to the Dirichlet problem for Poisson's equation combining the well known WOS method with use of the Poisson kernel as a probability density function. Using the Poisson kernel inside each WOS step, we avoid the need for detailed information about the Brownian trajectory inside the spherical domain. The Brownian trajectory is thus constructed as a series of discrete jumps using WOS with the source contribution inside each WOS step computed using the Poisson kernel.

Recently, we developed a modified WOS algorithm for solving the linearized Poisson-Boltzmann equation (LPBE) [7] in a domain Ω:

$$\Delta\psi(\mathbf{x}) = \kappa^2\psi(\mathbf{x}), \qquad \mathbf{x} \in \Omega, \tag{13}$$

$$\psi(\mathbf{x}) = \psi_0(\mathbf{x}), \qquad \mathbf{x} \in \partial\Omega. \tag{14}$$

Here, κ is called the inverse Debye length [14]. We used a survival probability, which was obtained by reinterpreting a weight function in a previously modified WOS method [4]. This survival probability enabled us to terminate some Brownian trajectories during WOS steps. This method can be combined with the method described in this paper to solve the Dirichlet boundary value problem for $\Delta\psi(\mathbf{x}) - \kappa^2\psi(\mathbf{x}) = -g(\mathbf{x})$. This will be the subject of a future study.

References

1. M. E. Müller. Some continuous Monte Carlo methods for the Dirichlet problem. *Ann. Math. Stat.*, 27:569–589, 1956.
2. K. K. Sabelfeld. *Monte Carlo Methods in Boundary Value Problems.* Springer-Verlag, Berlin, 1991.
3. A. Haji-Sheikh and E. M. Sparrow. The solution of heat conduction problems by probability methods. *Journal of Heat Transfer*, 89:121–131, 1967.
4. B. S. Elepov and G. A. Mihailov. The "Walk On Spheres" algorithm for the equation $\Delta u - cu = -g$. *Soviet Math. Dokl.*, 14:1276–1280, 1973.
5. T. E. Booth. Exact Monte Carlo solution of elliptic partial differential equations. *J. Comput. Phys.*, 39:396–404, 1981.
6. T. E. Booth. Regional Monte Carlo solution of elliptic partial differential equations. *J. Comput. Phys.*, 47:281–290, 1982.
7. C.-O. Hwang and M. Mascagni. Efficient modified "Walk On Spheres" algorithm for the linearized Poisson-Boltzmann equation. *Appl. Phys. Lett.*, 78(6):787–789, 2001.
8. J. M. DeLaurentis and L. A. Romero. A Monte Carlo method for Poisson's equation. *J. Comput. Phys.*, 90:123–139, 1990.
9. M. Freidlin. *Functional Integration and Partial Differential Equations.* Princeton University Press, Princeton, New Jersey, 1985.
10. K. L. Chung and Z. Zhao. *From Brownian Motion to Schrödinger's Equation.* Springer-Verlag, Berlin, 1995.
11. K. K. Sabelfeld. Integral and probabilistic representations for systems of elliptic equations. *Mathematical and Computer Modelling*, 23:111–129, 1996.
12. L. H. Zheng and Y. C. Chiew. Computer simulation of diffusion-controlled reactions in dispersions of spherical sinks. *J. Chem. Phys.*, 90(1):322–327, 1989.
13. S. Torquato and I. C. Kim. Efficient simulation technique to compute effective properties of hetergeneous media. *Appl. Phys. Lett.*, 55:1847–1849, 1989.
14. R. Ettelaie. Solutions of the linearized Poisson-Boltzmann equation through the use of random walk simulation method. *J. Chem. Phys.*, 103(9):3657–3667, 1995.

Relaxed Monte Carlo Linear Solver

Chih Jeng Kenneth Tan[1] and Vassil Alexandrov[2]

[1] School of Computer Science
The Queen's University of Belfast
Belfast BT7 1NN
Northern Ireland
United Kingdom
cjtan@acm.org
[2] Department of Computer Science
The University of Reading
Reading RG6 6AY
United Kingdom
v.n.alexandrov@rdg.ac.uk

Abstract. The problem of solving systems of linear algebraic equations by parallel Monte Carlo numerical methods is considered. A parallel Monte Carlo method with relaxation is presented. This is a report of a research in progress, showing the effectiveness of this algorithm. Theoretical justification of this algorithm and numerical experiments are presented. The algorithms were implemented on a cluster of workstations using MPI.
Keyword: Monte Carlo method, Linear solver, Systems of linear algebraic equations, Parallel algorithms.

1 Introduction

One of the more common numerical computation task is that of solving large systems of linear algebraic equations

$$Ax = b \tag{1}$$

where $A \in \mathbb{R}^{n \times n}$ and $x, b \in \mathbb{R}^n$. A great multitude of algorithms exist for solving Equation 1. They typically fall under one of the following classes: direct methods, iterative methods, and Monte Carlo methods. Direct methods are particularly favorable for dense A with relatively small n. When A is sparse, iterative methods are preferred when the desired precision is high and n is relatively small. When n is large and the required precision is relatively low, Monte Carlo methods have been proven to be very useful [6,4,15,1].

As a rule, Monte Carlo methods are not competitive with classical numerical methods for solving systems of linear algebraic equations, if the required precision is high [13].

In Monte Carlo methods, statistical estimates for the components of the solution vector x are obtained by performing random sampling of a certain random

V.N. Alexandrov et al. (Eds.): ICCS 2001, LNCS 2073, pp. 1289–1297, 2001.
© Springer-Verlag Berlin Heidelberg 2001

variable whose mathematical expectation is the desired solution [14,18]. These techniques are based on that proposed by von Neumann and Ulam, extended by Forsythe and Liebler [13,9].

Classical methods such as non-pivoting Gaussian Elimination or Gauss-Jordan methods require $\mathcal{O}(n^3)$ steps for a $n \times n$ square matrix [2]. In contrast, to compute the full solution vector using Monte Carlo the total number of steps required is $\mathcal{O}(nNT)$, where N is the number of chains and T is the chain length, both quantities independent of n and bounded [1]. Also, if only a few components of x are required, they can be computed without having to compute the full solution vector. This is a clear advantage of Monte Carlo methods, compared to their direct or iterative counterpart.

In addition, even though Monte Carlo methods do not yield better solutions than direct or iterative numerical methods for solving systems of linear algebraic equations as in Equation 1, they are more efficient for large n. Also, Monte Carlo methods have been known for their embarrassingly parallel nature. Parallelizing Monte Carlo methods in a coarse grained manner is very often straightforward. This characteristic of Monte Carlo methods has been noted even in 1949, by Metropolis and Ulam [12].

2 Stochastic Methods for Solving Systems of Linear Algebraic Equations

Consider a matrix $A \in \mathbb{R}^{n \times n}$ and a vector $x \in \mathbb{R}^{n \times 1}$. Further, A can be considered as a linear operator $A\left[\mathbb{R}^{n \times 1} \rightarrow \mathbb{R}^{n \times 1}\right]$, so that the linear transformation

$$Ax \in \mathbb{R}^{n \times 1} \tag{2}$$

defines a new vector in $\mathbb{R}^{n \times 1}$.

The linear transformation in Equation 2 is used in iterative Monte Carlo algorithms, and the linear transformation in Equation 2 is also known as the iteration. This algebraic transform plays a fundamental role in iterative Monte Carlo algorithms.

In the problem of solving systems of linear algebraic equations, the linear transformation in Equation 2 defines a new vector $b \in \mathbb{R}^{n \times 1}$:

$$Ax = b, \tag{3}$$

where A and b are known, and the unknown solution vector x is to be solved for. This is a problem often encountered as subproblems on in various applications such as solution of differential equations, least squares solutions, amongst others.

It is known that system of linear algebraic equation given by Equation 3, can be rewritten in the following iterative form [2,18,4]:

$$x = Lx + b, \tag{4}$$

where

$$(I - L) = A. \tag{5}$$

Assuming that $\|L\| < 1$, and $x^0 \equiv 0$, the von Neumann series converges and the equation

$$\lim_{k\to\infty} x^{(k)} = \lim_{k\to\infty} \sum L^m b = (I - L)^{-1} b = A^{-1} b = x \tag{6}$$

holds.

Suppose now $\{s_1, s_2, \ldots, s_n\}$ is a finite discrete Markov chains with n states. At each discrete time $t = 0, 1, \ldots, N$, a chain S of length T is generated:

$$k_0 \to k_1 \to \ldots \to k_j \to \ldots \to k_T$$

with $k_j \in \{s_1, s_2, \ldots, s_n\}$ for $j = 1, \ldots, T$.

Define the probability that the chain starts in state s_α,

$$\mathbf{P}\left[k_0 = s_\alpha\right] = p_\alpha \tag{7}$$

and the transition probability to state s_β from state s_α

$$\mathbf{P}\left[k_j = s_\beta | k_{j-1} = s_\alpha\right] = p_{\alpha\beta} \tag{8}$$

for $\alpha = 1, \ldots, n$ and $\beta = 1, \ldots, n$.

The probabilities $p_{\alpha\beta}$ thus define the transition matrix P. The distribution $(p_1, \ldots, p_n)^T$ is said to be acceptable to vector h, and similarly that the distribution $p_{\alpha\beta}$ is acceptable to L, if [14]

$$\begin{cases} p_\alpha > 0 \text{ when } h_\alpha \neq 0 \\ p_\alpha \geq 0 \text{ when } h_\alpha = 0 \end{cases} \text{ and } \begin{cases} p_{\alpha\beta} > 0 \text{ when } l_{\alpha\beta} \neq 0 \\ p_{\alpha\beta} \geq 0 \text{ when } l_{\alpha\beta} = 0 \end{cases} \tag{9}$$

Define the random variables W_j according to the recursion

$$W_j = W_{j-1} \frac{l_{k_{j-1}k_j}}{p_{k_{j-1}k_j}},\ W_0 \equiv 1 \tag{10}$$

The random variables W_j can also be considered as weights on the Markov chain.

Also, define the random variable

$$\eta_T(h) = \frac{h_{k_0}}{p_{k_0}} \sum_{j=0}^{T-1} W_j b_{k_j}. \tag{11}$$

From Equation 6, the limit of $\mathbf{M}\left[\eta_T(h)\right]$, the mathematical expectation of $\eta_T(h)$ is

$$\mathbf{M}\left[\eta_T(h)\right] = \left\langle h, \sum_{m=0}^{T-1} L^m b \right\rangle = \left\langle h, x^{(T)} \right\rangle \Rightarrow \lim_{T\to\infty} \mathbf{M}\left[\eta_T(h)\right] = \langle h, x \rangle \tag{12}$$

Knowing this, one can find an unbiased estimator of $\mathbf{M}\left[\eta_\infty(h)\right]$ in the form

$$\theta_N = \frac{1}{N} \sum_{m=0}^{N-1} \eta_\infty(h) \tag{13}$$

Consider functions $h \equiv h^j = (0, 0, \ldots, 1, \ldots, 0)$, where $h_i^j = \delta_i^j$ is the Kronecker delta. Then

$$\langle h, x \rangle = \sum_{i=0}^{n-1} h_i^j x_i = x_j. \tag{14}$$

It follows that an approximation to x can be obtained by calculating the average for each component of every Markov chain

$$x_j \approx \frac{1}{N} \sum_{m=0}^{N-1} \theta_T^m \left[h^j\right] . \tag{15}$$

In summary, N independent Markov chains of length T is generated and $\eta_T(h)$ is calculated for each path. Finally, the j-th component of x is estimated as the average of every j-th component of each chain.

3 Minimal Probable Error

Let I be any functional to be estimated by Monte Carlo method, θ be the estimator, and n be the number of trials. The probable error for the usual Monte Carlo method is defined as [14]:

$$\mathbf{P}\left[|I - \theta| \geq r\right] = \frac{1}{2} = \mathbf{P}\left[|I - \theta| \leq r\right]. \tag{16}$$

Equation (16) does not take into consideration any additional a priori information regarding the regularity of the solution.

If the standard deviation $(\mathbf{D}\left[\theta\right])^{\frac{1}{2}}$ is bounded, then the Central Limit Theorem holds, and

$$\mathbf{P}\left[|I - \theta| \leq x \left(\frac{\mathbf{D}\left[\theta\right]}{n}\right)^{\frac{1}{2}}\right] \approx \Phi(x). \tag{17}$$

Since $\Phi(0.6745) \approx \frac{1}{2}$, it is obvious that the probable error is

$$r \approx 0.6745 \left(\mathbf{D}\left[\theta\right]\right)^{\frac{1}{2}} . \tag{18}$$

Therefore, if the number of Markov chains N increases the error bound decreases. Also the error bound decreases if the variance of the random variable θ decreases.

This leads to the definition of almost optimal transition frequency for Monte Carlo methods. The idea is to find a transition matrix P that minimize the second moment of the estimator. This is achieved by choosing the probability proportional to the $|l_{\alpha\beta}|$ [6]. The corresponding almost optimal initial density vector, and similarly the transition density matrix $P = \{p_{\alpha\beta}\}_{\alpha,\beta=1}^{n}$ is then called the almost optimal density matrix.

4 Parameter Estimation

The transition matrix P is chosen with elements $p_{\alpha\beta} = \frac{|l_{\alpha\beta}|}{\sum_{\beta}|l_{\alpha\beta}|}$ for $\alpha, \beta = 1, 2, ..., n$. In practice the length of the Markov chain must be finite, and is terminated when $|W_j b_{k_j}| < \delta$, for some small value δ [14]. Since

$$|W_j b_{k_j}| = \left| \frac{l_{\alpha_0\alpha_1} \cdots l_{\alpha_{j-1}\alpha_j}}{\frac{|l_{\alpha_0\alpha_1}|}{\|L\|} \cdots \frac{|l_{\alpha_{j-1}\alpha_j}|}{\|L\|}} \right| |b_{k_j}| = \|L\|^i \|b\| < \delta\,, \tag{19}$$

it follows that

$$T = j \leq \frac{\log\left(\frac{\delta}{\|b\|}\right)}{\log\|L\|} \tag{20}$$

and

$$\mathbf{D}\left[\eta_T(h)\right] \leq \mathbf{M}\left[\eta_T^2\right] = \frac{\|b\|^2}{\left(1-\|L\|\right)^2} \leq \frac{1}{\left(1-\|L\|\right)^2}. \tag{21}$$

According to the Central Limit Theorem,

$$N \geq \left(\frac{0.6745}{\epsilon}\right)^2 \frac{1}{\left(1-\|L\|\right)^2} \tag{22}$$

is a lower bound on N.

5 Relaxed Monte Carlo Method

Here, the Relaxed Monte Carlo method is defined. Consider a matrix $E \in \mathbb{R}^{n\times n}$. Multiply matrix E to both left and right sides of Equation 3, so that

$$EAx = Eb. \tag{23}$$

It is then possible to define an iteration matrix L_r,

$$L_r = I - EA, \tag{24}$$

similar to the iteration matrix L Equation 5. It then follows that

$$x = L_r x + f, \tag{25}$$

where

$$f = Eb. \tag{26}$$

The corresponding von Neumann series converges and

$$x^{(k+1)} = \left(I + L_r + L_r^2 + \cdots + L_r^k\right) f = \sum_{m=0}^{k} L_r^m f \text{ where } L_r^0 \equiv I, \tag{27}$$

where

$$\lim_{k\to\infty} x^{(k)} = \lim_{k\to\infty} \sum L_r^m f = (I - L_r)^{-1} f = (EA)^{-1} f = x. \tag{28}$$

Define E as a (diagonal) matrix such that

$$e_{ij} = \begin{cases} \frac{\gamma}{a_{ij}}, \gamma \in (0,1], & \text{if } i = j \\ 0, & \text{if } i \neq j \end{cases}. \tag{29}$$

The parameter γ is chosen such that it minimizes the norm of L in order to accelerate the convergence. This is similar to the relaxed successive approximation iterative method with the relaxation parameter γ [5]. Similar approach was presented and discussed by Dimov et al. [5] for iterative Monte Carlo method for solving inverse matrix problems, where the matrices were diagonally dominant. The parameter γ was then changed dynamically during the computation. In contrast, the parameter γ chosen in this case is based on a priori information, so that matrix norm is reduced, preferably to less than 0.5.

Furthermore, a set of parameters, $\{\gamma_i\}_{i=0}^{n}$ can be used in place of a single γ value, to give a desirable norm in each row of L_r. Such a choice will also result in a matrix L_r which is more balanced, in terms of its row norms, $\|L_{ri}\|$.

Following the arguments of Faddeev and Faddeeva [7,8], this Relaxed Monte Carlo method will converge if

$$\gamma_i < \frac{2}{\|A_i\|}, \tag{30}$$

where $\|A_i\|$ is the row norm of the given matrix A. This approach is equally effective for both diagonally dominant and non-diagonally dominant matrices. This is also corroborated by the numerical experiments conducted.

It is obvious that the Relaxed Monte Carlo method can be used in conjunction with either the almost optimal Monte Carlo method or the Monte Carlo method with chain reduction and optimization [16,17,3]. In any case, since the Relaxed Monte Carlo method can be used to reduce the norm of a matrix to a specified value, it can always be used to the effect of accelerating the convergence of the Monte Carlo method in general.

6 Numerical Experiments

A parallel version of the Relaxed Monte Carlo algorithm was developed using Message Passing Interface (MPI) [11,10]. Version 1.2.0 of the MPICH implementation of the Message Passing Interface was used. As the programs were written in C, the C interface of MPI was the natural choice interface.

Tables 1 and 2, show the results for experiments with the Relaxed Monte Carlo method. The matrices used in these experiments were dense (general) randomly populated matrices, with a specified norm. The stochastic error, ϵ, and the deterministic error parameters were both set to 0.01. The `PLFG` parallel pseudo-random number generator [17] was used as the source of randomness for the experiments conducted.

Table 1. Relaxed Monte Carlo method with `PLFG`, using 10 processors, on a DEC Alpha XP1000 cluster.

Data set	Norm	Solution time (sec.)	RMS error	No. chains
100-A1	0.5	0.139	4.76872e-02	454900
100-A2	0.6	0.122	4.77279e-02	454900
100-A3	0.7	0.124	4.78072e-02	454900
100-A4	0.8	0.127	4.77361e-02	454900
100-A5	0.5	0.137	3.17641e-02	454900
100-A6	0.6	0.124	3.17909e-02	454900
100-A7	0.7	0.124	3.17811e-02	454900
100-A8	0.8	0.119	3.17819e-02	454900
100-B1	0.5	0.123	3.87367e-02	454900
100-B2	0.6	0.126	3.87241e-02	454900
100-B3	0.7	0.134	3.88647e-02	454900
100-B4	0.8	0.125	3.88836e-02	454900
100-B5	0.5	0.121	2.57130e-02	454900
100-B6	0.6	0.119	2.57748e-02	454900
100-B7	0.7	0.120	2.57847e-02	454900
100-B8	0.8	0.126	2.57323e-02	454900

The time to solution given in the tables is the actual computation time, in seconds. The time taken to load the data is not taken into account, since for many computational science problems, the data are created on the nodes [1].

7 Acknowledgment

We would like to thank M. Isabel Casas Villalba from Norkom Technologies, Ireland for the fruitful discussions and the MACI project at the University of Calgary, Canada, for their support, providing part of the computational resources used.

References

[1] Alexandrov, V. N. Efficient Parallel Monte Carlo Methods for Matrix Computations. *Mathematics and Computers in Simulation 47*, 2 – 5 (1998), 113 – 122.

[2] Bertsekas, D. P., and Tsitsiklis, J. N. *Parallel and Distributed Computation: Numerical Methods*. Athena Scientific, 1997.

[3] Casas Villalba, M. I., and Tan, C. J. K. Efficient Monte Carlo Linear Solver with Chain Reduction and Optimization Using `PLFG`. (To be published.), 2000.

[4] Dimov, I. Monte Carlo Algorithms for Linear Problems. In *Lecture Notes of the 9th. International Summer School on Probability Theory and Mathematical Statistics* (1998), N. M. Yanev, Ed., SCT Publishing, pp. 51 – 71.

Table 2. Relaxed Monte Carlo method with `PLFG`, using 10 processors, on a DEC Alpha XP1000 cluster.

Data set	Norm	Solution time (sec.)	RMS error	No. chains
1000-A1	0.5	7.764	1.91422e-02	2274000
1000-A2	0.6	7.973	1.92253e-02	2274000
1000-A3	0.7	7.996	1.93224e-02	2274000
1000-A4	0.8	7.865	1.91973e-02	2274000
1000-A5	0.5	7.743	1.27150e-02	2274000
1000-A6	0.6	7.691	1.27490e-02	2274000
1000-A7	0.7	7.809	1.27353e-02	2274000
1000-A8	0.8	7.701	1.27458e-02	2274000
1000-B1	0.5	7.591	1.96256e-02	2274000
1000-B2	0.6	7.587	1.97056e-02	2274000
1000-B3	0.7	7.563	1.96414e-02	2274000
1000-B4	0.8	7.602	1.96158e-02	2274000
1000-B5	0.5	7.147	1.29432e-02	2274000
1000-B6	0.6	7.545	1.30017e-02	2274000
1000-B7	0.7	7.541	1.31470e-02	2274000
1000-B8	0.8	7.114	1.28813e-02	2274000

[5] Dimov, I., Dimov, T., and Gurov, T. A new iterative monte carlo approach for inverse matrix problem. *Journal of Computational and Applied Mathematics 4*, 1 (1998), 33 – 52.

[6] Dimov, I. T. Minimization of the Probable Error for some Monte Carlo Methods. In *Mathematical Modelling and Scientific Computations* (1991), I. T. Dimov, A. S. Andreev, S. M. Markov, and S. Ullrich, Eds., Publication House of the Bulgarian Academy of Science, pp. 159 – 170.

[7] Faddeev, D. K., and Faddeeva, V. N. *Computational Methods of Linear Algebra*. Nauka, Moscow, 1960. (In Russian.).

[8] Faddeeva, V. N. *Computational Methods of Linear Algebra*. Nauka, Moscow, 1950. (In Russian.).

[9] Forsythe, S. E., and Liebler, R. A. Matrix Inversion by a Monte Carlo Method. *Mathematical Tables and Other Aids to Computation 4* (1950), 127 – 129.

[10] Message Passing Interface Forum. *MPI: A Message-Passing Interface Standard*, 1.1 ed., June 1995.

[11] Message Passing Interface Forum. *MPI-2: Extensions to the Message-Passing Interface*, 2.0 ed., 1997.

[12] Metropolis, N., and Ulam, S. The monte carlo method. *Journal of the American Statistical Association 44*, 247 (1949), 335 – 341.

[13] Rubinstein, R. Y. *Simulation and the Monte Carlo Method*. John Wiley and Sons, 1981.

[14] Sobol', I. M. *Monte Carlo Numerical Methods*. Nauka, Moscow, 1973. (In Russian.).

[15] Tan, C. J. K., and Blais, J. A. R. PLFG: A Highly Scalable Parallel Pseudo-random Number Generator for Monte Carlo Simulations. In *High Performance Computing and Networking, Proceedings of the 8th. International Conference on High Performance Computing and Networking Europe* (2000), M. Bubak, H. Afsarmanesh, R. Williams, and B. Hertzberger, Eds., vol. 1823 of *Lecture Notes in Computer Science*, Springer-Verlag, pp. 127 – 135.

[16] Tan, C. J. K., Casas Villalba, M. I., and Alexandrov, V. An Improved Monte Carlo Linear Solver Algorithm. (To be published.), 2001.

[17] Tan, C. J. K., Casas Villalba, M. I., and Alexandrov, V. N. Monte Carlo Method for Solution of Linear Algebraic Equations with Chain Reduction and Optimization Using PLFG. In *Proceedings of the 2000 SGI Users' Conference* (2000), M. Bubak, J. Mościński, and M. Noga, Eds., Academic Computing Center, CYFRONET, AGH, Poland, pp. 400 – 408.

[18] Westlake, J. R. *A Handbook of Numerical Matrix Inversion and Solution of Linear Equations.* John Wiley and Sons, 1968.

Author Index

Aarts, Lucie P., II, 181
Abdulmuin, Mohd. Zaki, II, 198
Abraham, Ajith, II, 171, II, 235, II, 337
Adamyan, H.H., II, 1041
Addison, Cliff, I, 3
Agarwal, D.A., I, 316
Ahtiwash, Otman M., II, 198
Akbar, Md. Mostofa, II, 659
Akhmetov, Dauren, I, 284
Akin, Erhan, II, 272
Albada, G.D. van, II, 883
Alberto, Pedro, II, 95
Alexandrov, Vassil, I, 1289
Alias, Norma, I, 918
Allred, Ken, II, 550
Anderson, David R., II, 550
Anderson, John, I, 175
Andrianov, A.N., I, 502
Arickx, Frans, I, 423
Arrighi, William J., II, 158
Arslan, Ahmet, II, 190
Atiqullah, Mir M., II, 669
d'Auriol, Brian J., II, 701

Baden, Scott B., I, 785
Bader, David A., II, 1012
Bajze, Željko, II, 680
Baker, A. Jerry, I, 336
Baker, M. Pauline, II, 718
Baklanov, Alexander, II, 57
Balaton, Zoltán, I, 253
Banavar, Jayanth R., I, 551
Baracca, M.C., II, 99
Barber, Michael J., II, 958, II, 996
Barth, Eric, II, 1065
Bates, Charles, II, 507
Baum, Joseph D., I, 1087
Beckner, Vince E., I, 1117
Bekker, Henk, I, 619
Bell, David, II, 245
Belloni, Mario, I, 1061
Berket, K., I, 316
Bernal, Javier, I, 629
Berzins, Martin, II, 67
Bespamyatnikh, Sergei, I, 633
Bettge, Thomas, I, 149
Bhandarkar, Milind A., II, 108
Bhattacharya, Amitava, I, 640
Bhattacharya, Maumita, II, 1031
Bian, Xindi, I, 195
Bilardi, Gianfranco, II, 579
Billock, Joseph Greg, II, 208
Bilmes, Jeff, I, 117
Bischof, Christian H., I, 795
Blais, J.A. Rod, II, 3
Blanchette, Mathieu, II, 1003
Boatz, Jerry, I, 1108
Boctor, Emad, II, 13
Bogdanov, Alexander V., I, 447, I, 473, II, 965
Bond, Steve D., II, 1066
Bonitz, M., I, 1272
Boukhanovsky, A.V., I, 453, I, 463
Breg, Fabian, I, 223
Broeckhove, Jan, I, 423
Brooks, Malcolm, II, 390
Brown, William R., II, 550
Brunst, Holger, II, 751
Bücker, H. Martin, I, 795
Bukovsky, Antonin, I, 355
Burgin, Mark, II, 728
Burroughs, Ellis E., II, 550

Cáceres, E.N., II, 638
Carenini, Guiseppe, I, 959
Carofiglio, Valeria, I, 1019
Carroll, Steven, I, 223
Carter, Larry, I, 81, I, 137
Carvalho, João Paulo, II, 217
Casas, Claudia V., II, 701
Chandramohan, Srividya, I, 404
Chang, Dar-Jen, II, 13
Chang, Kyung-Ah, I, 413, I, 433
Chappell, M.A., I, 1237
Charman, Charles, I, 1087

Chassin de Kergommeaux, J., II, 831
Chatterjee, Siddhartha, I, 107
Chen, Jianer, II, 609
Chen, Xiaodong, II, 485
Cheung, W.L., I, 862
Chikkappaiah, Pramod Kumar, II, 701
Chin, Jr., George, I, 159
Choi, Hyung-Il, II, 37, II, 148
Choi, Jaeyoung, I, 802, II, 148
Choo, Hyunseung, II, 912
Christiaens, Mark, II, 761, II, 851
Christian, Wolfgang, I, 1061
Christie, Michael, I, 1170
Chung, Tai M., II, 912
Çinar, Ahmet, II, 190
Clai, G., II, 99
Clardy, Tim, II, 550
Cook, David M., I, 1074
Corbin, Jim, II, 476
Cornelis, Chris, II, 221
Craig, Anthony, I, 149
Craig, David, I, 223
Crocchianti, Stefano, I, 567
Cucos, Laurentiu, II, 118
Cunha, José C., II, 821
Curington, Ian, II, 711

Dabdub, Donald, II, 77
Daescu, Ovidiu, I, 649
D'Agosto, Giuseppina, I, 567
Dalgalarrondo, André, II, 327
Dancy, Melissa, I, 1061
Dawsey, Shanda K., II, 952
De Bosschere, Koen, II, 761, II, 851
Debenham, John K., I, 1219
Dediu, Adrian Horia, II, 419
Degtyarev, A.B., I, 453, I, 463, II, 965
Dehne, Frank, II, 589
Dellen, Babette K., II, 996
Demmel, James W., I, 31, I, 117
Deng, X., II, 648
Dennen, Kevin, II, 550
DeTar, Carleton, I, 1176
Dey, Tamal K., I, 658
Domínguez, J.J., II, 318
Doncker, Elise de, II, 118
Dongarra, Jack J., I, 41, I, 355
Dooley, Laurence S., II, 281
Doran, Tony, I, 812
Drake, Marsha, II, 558
Draper, L. Susan, II, 701
Drkošová, Jitka, II, 986
Drummond, L.A., I, 31
Duan, Yun-Bo, II, 893
Durak, Lora J., II, 976

Eavis, Todd, II, 589
Eddings, Eric, II, 485
Efimkin, K.N., I, 502
Ercan, M. Fikret, I, 61, I, 862
Espen, Peter K., II, 158
Esper, Ammar J., II, 701
Evans, Gwynne, I, 852

Fagg, Graham E., I, 41, I, 355
Fantozzi, Carlo, II, 579
Farris, Jeremy R., II, 558
Feather, B.K., I, 1237
Ferraiolo, David F., II, 494
Ferrante, Jeanne, I, 81, I, 137
Ferrigno, Giancarlo, II, 23
Fiedler, Armin, I, 969
Filinov, V., I, 1272
Flanery, Raymond E., II, 871
Flitman, Andrew, II, 447
Fogelson, Aaron L., I, 1176
Forlani, Christian, II, 23
Fortov, V., I, 1272
Foster, Ian T., I, 175, I, 185
Frechard, F., I, 531
Freeman, Walter J., II, 231
Frincke, Deborah A., II, 494
Frumkin, Michael, II, 771
Fung, Ming, II, 447
Fung, Yu-Fai, I, 61, I, 862

Gallagher, Ray, I, 812, I, 1253
Gálvez, Akemi, I, 698
Gavenko, Serge V., I, 979
Gavrilova, M.L., I, 663, I, 673, I, 748

Geijn, Robert A. van de, I, 51
Geoffray, P., I, 233
Gevorkyan, Ashot S., I, 447, I, 473
Ghosh, Subir Kumar, I, 640
Gidaspov, Vladimir, I, 511
Giesen, Joachim, I, 658
Gilbert, Michael A., I, 989
Ginsberg, Myron, I, 1189
Giordano, N., I, 1041
Glasner, Christian, II, 781
Glimm, James, I, 5
Gorbachev, Yu. E., I, 483
Gordon, Mark S., I, 1108
Gossage, Brett N., I, 1209, II, 531, II, 540
Gould, Harvey, I, 1031
Gracio, Debbie, I, 159
Grama, Ananth, II, 599
Grasso, Floriana, I, 999
Gray, Paul A., I, 307, I, 404
Green, Nancy, I, 1009
Grimshaw, Andrew S., I, 273
Guerrero, F., II, 318
Gunnels, John A., I, 51
Guo, Gongde, II, 245
Guo, Jing, I, 185
Gupta, Anshul, I, 823

Hammes-Shiffer, Sharon, I, 1108
Hardy, David J., II, 1067
Hart, Delbert, II, 791
Harter, Derek, II, 300
Haszpra, Lászlo, II, 67
Hedin, Johan, I, 1170
Heiland, Randy W., II, 718
Henry, Greg M., I, 51
Hermse, C.G.M., I, 531
Hilbers, P.A.J., I, 541
Hitt, Ray T., II, 158
Hlavacs, Helmut, I, 243
Ho, T.K, I, .862
Höfinger, Siegfried, II, 801
Hoeflinger, Jay, II, 108
Hoekstra, A.G., I, 518
Hon, Man Chung, I, 683
Hong, Chung-Seong, I, 693
Hong, Seok-Yong, I, 693
Hoppe, Hans-Christian, II, 751
Horacek, Helmut, I, 969
Housni, Ahmed, I, 294
Hu, Hong, I, 1137
Hu, Vincent C., II, 494
Huber, Valentina, I, 560
Hübler, Alfred W., II, 976
Hügl, Roland, II, 781
Humphrey, Marty A., I, 273
Hunzelmann, Gunnar T., II, 841
Hwang, Chi-Ok, I, 1282

Iglesias, Andrés, I, 698
Ignatiev, A.A., I, 483
Im, Eun-Jin, I, 127
Ip, H., II, 648
Ishi, Naohiro, II, 138
Itabashi, Shingo, II, 138
Iwata, Kazunori, II, 138

Jacob, Robert L., I, 175, I, 185
Jacobson, Mark Z., II, 1060
James, Rodney, I, 149
Janardan, Ravi, I, 683
Janoski, Guadalupe L., II, 253
Jansen, A.P.J., I, 531
Jaun, André, I, 1170
Jeong, Chang-Sung, II, 44
Jin, Haoqiang, II, 771
Johnson, Christopher R., I, 6, I, 1176
Johnson, James B., II, 515
Johnson, Thomas, I, 1170
Johnsson, S. Lennart, I, 71
Jonsson, Lars-Erik, I, 1170
Joppich, Wolfgang, I, 492
Joseph, K. Babu, II, 400
Joyner, Tom, II, 521
Juliano, Benjoe A., II, 943

Kacprzyk, Janusz, II, 263
Kacsuk, Péter, I, 7, I, 253
Kalé, L.V., II, 108
Kamel, K., II, 13
Kämpke, Thomas, I, 708
Karakose, Mehmet, II, 272

Karl, Wolfgang, II, 27, II, 861
Karmakar, Gour C., II, 281
Karplus, Walter, II, 728
Kaugars, Karlis, II, 118
Kaya, Mehmet, II, 272
Kennedy, Ken, I, 8
Kerre, Etienne E., II, 221
Khan, Shahadat, II, 659
Khoo, Wee Sng, II, 291
Khuri, Sami, II, 689
Kim, Deok-Soo, I, 718, I, 728
Kim, Donguk, I, 718, I, 728
Kim, Gye-Young, II, 37
Kim, Tai-Yun, I, 413, I, 433
King, Harry F., I, 911
Knight, Claire, II, 470
Kobayashi, Kei, I, 738
Koch, Christof, II, 208
Kohl, James A., II, 871
Komornicki, Andrew, I, 28
Koo, Han-Suh, II, 44
Koplik, Joel, I, 551
Korneva, Alexandra A., I, 832
Kozma, Robert, II, 300
Kraeft, W.-D., I, 1272
Kraemer, Eileen, II, 791
Kranzlmüller, Dieter, II, 781, II, 811
Kremp, D., I, 1272
Kriz, John, II, 521
Krušina, Pavel, II, 935
Kryuchkyan, G. Yu, II, 1041
Krzhizhanovskaya, V.V., I, 483
Kulikov, Gennady Yu., I, 832
Kumar, Vipin, II, 599
Kurzyniec, Dawid, I, 375
Kvasnicka, Dieter F., I, 243

Labbé, C., II, 831
Laganà, Antonio, I, 567
Lagzi, István, II, 67
Laird, Brian B., II, 1068
Landau, Rubin H., I, 1051
Lang, Bruno, I, 795
Laporte, Yan, II, 366
Larget, Bret, II, 1022
Lari-Lavassani, Ali, I, 597
Larson, Jay Walter, I, 185, I, 204
Law, K., II, 648
L'Ecuyer, Pierre, I, 9, I, 607
Lee, Byung-Rae, I, 413, I, 433
Lee, Hyun-Chan, I, 693
Lee, Joong-Jae, II, 37
Leimkuhler, Benedict J., II, 1069
Lemieux, Christiane, I, 607
LeSueur, Kenneth G., II, 531, II, 540, II, 550
Leung, L. Ruby, I, 159, I, 195
Levashov, V. Yu., I, 502
Li, J., II, 648
Li, Xin Kai, I, 852
Lijewski, Mike J., I, 1117
Lin, Chih-Yang, II, 429
Lin, Feng-Tse, II, 409
Lin, Zhiping, II, 310
Lindstrom, Gary, II, 485
Liu, Damon, II, 728
Loader, R.J., I, 326, I, 385
Löhner, Rainald, I, 1087
López, Jorge, II, 701
Lourenço, João, II, 821
Lovas, Robert, I, 263
Lozano, S., II, 318
Lu, Mi, I, 884
Luchnikov, V.A., I, 748
Lukkien, J.J., I, 531
Luo, Hong, I, 1087
Luzeaux, Dominique, II, 327

Ma, K.T., II, 439
Mahanti, P.K., II, 337
Maltrud, Mathew E., I, 1098
Maneewongvatana, Songrit, I, 842
Manning, Eric G., II, 659
Manvelyan, S.B., II, 1041
Manzardo, Mark A., II, 531, II, 540
Margrave, Gary F., I, 874
Mark, Andrew, I, 1199
Markvoort, A.J., I, 541
Mascagni, Michael, I, 1282
Maslowski, Wieslaw, I, 1098
Matzke, Robb P., II, 158
McClean, Julie L., I, 1098

McCoy, Anne B., II, 893
McCrindle, Rachel Jane, II, 459, II, 738
McGraner, Greg, II, 521
McNamara, Sean, I, 551
Mechoso, C.R., I, 31
Medvedev, N.N., I, 748
Mehra, Avichal, II, 476
Melliar-Smith, P.M., I, 316
Meng, Sha, I, 852
Menon, Anil, II, 922
Menon, Suresh, I, 1127
Mestreau, Eric L., I, 1087
Mészáros, Róbert, II, 67
Meyer auf der Heide, Friedhelm, II, 628
Mi, Yanpeng, I, 874
Michael, T.S., I, 753
Michalakes, John G., I, 195
Mierendorff, Herrmann, I, 492
Migliardi, Mauro, I, 345, I, 367
Miller, Mark C., II, 158
Mills Strout, Michelle, I, 137
Mirković, Dragan, I, 71
Mishra, S., II, 128
Mitchell, Nick, I, 81
Mohades, Ali, I, 763
Mohan, Ram, I, 1199
Moisa, Trandafir, II, 419
Molakaseema, Rajesh, II, 701
Moreira, José E., I, 10
Moret, Bernard M.E., II, 1012
Moser, L.E., I, 316
Mount, David M., I, 842
Moura, José M.F., I, 97
Mun, Youngsong, II, 148
Munro, Malcolm, II, 470
Muruzábal, Jorge, II, 346
Muslaev, Alexander, I, 511

Nagel, Wolfgang E., II, 751
Nath, Baikunth, II, 171, II, 337, II, 1031
Natrajan, Anand, I, 273
Nazaryan, Karen M., II, 356
Nechaev, Yu. I., I, 453, II, 965
Neruda, Roman, II, 935, II, 986
Nguyen, Khoi, II, 77
Nikolopoulos, Dimitrios S., I, 223
Nkambou, Roger, II, 366
Nogueira, Fernando, II, 95

Offord, Chetan, II, 680
Oguz, Ceyda, I, 61
Oh, Eunseuk, II, 609
Olive, V., II, 831
Oliveira Stein, B. de, II, 831
Oliver, Carl Edward, I, 4
Ontanu, Dan, II, 419
Ornelli, P., II, 99
Ottogalli, F.-G., II, 831

Pachter, Ruth, I, 1108
Pacifici, Leonardo, I, 567
Park, Kiheon, II, 912
Park, Koohyun, I, 693
Pascoe, J.S., I, 307, I, 326, I, 385
Pelessone, Daniele, I, 1087
Penenko, Vladimir, II, 57
Peng, Dongming, I, 884
Persson, Mikael, I, 1170
Petrová, Zuzana, II, 935
Pham, C., I, 233
Philip, Ninan Sajith, II, 400
Piermarini, Valentina, I, 567
Pietracaprina, Andrea, II, 579
Pinciu, Val, I, 753
Pino, R., I, 541
Podhorszki, Norbert, I, 253
Polychronopoulos, Constantine, I, 223
Port-Agel, Fernando, II, 1062
Primbs, James A., I, 579
Protopopov, V. Kh., I, 483
Prylli, L., I, 233
Psaltis, Demetri, II, 208
Pucci, Geppino, II, 579
Puig-Pey, Jaime, I, 698
Püschel, Markus, I, 97
Pyun, Soo Bum, I, 928

Qin, Qiao, II, 1062

Rahman, Syed M., II, 281

Rajabi, Mohammad A., II, 3
Rakoto-Ravalontsalama, N., I, 1228
Ramachandran, Vijaya, II, 619
Rangarajan, Govindan, I, 894
Ranka, Sanjay, II, 599
Rao, S.S., II, 669
Rasch, Arno, I, 795
Rau-Chaplin, Andrew, II, 589
Ray, Jerry A., II, 515
Razzazi, Mohammadreza, I, 763
Reed, Chris, I, 999
Reinefeld, Alexander, II, 569
Reitinger, Bernhard, II, 781
Rendleman, Charles, A., I, 1117
Renner, R.S., II, 943, II, 952
Reus, James F., II, 158
Reussner, Ralf H., II, 841
Risch, Jakob W., I, 795
Roberts, J., I, 1041
Roberts, Patrick, II, 550
Robinson, H., I, 31
Rocha, Humberto, II, 95
Rodionov, Alexey S., II, 912
Roerdink, Jos B.T.M., I, 619
Rokne, J., I, 673
Ronsse, Michiel, II, 851
Rose, John, I, 937
Rosis, Fiorella de, I, 1019
Rozhkov, V.A., I, 463
Rozovski, Peter, I, 511
Ryu, Joonghyun, I, 718, I, 728

Saad, A., II, 13
Sachidanand, Minita, I, 894
Saenz, René, II, 701
Sahimi, Mohd Salleh, I, 918
Sankaran, Vaidyanathan, I, 1127
Saratchandran, P., II, 291
Sarkar, Sudeep, I, 640
Sarofim, Adel F., II, 485
Satpathy, M., II, 128
Schafer, Chad, I, 175
Schevtschenko, I.V., I, 904
Schintke, Florian, II, 569
Schoof, Larry A., II, 158
Schuchardt, Karen, I, 159
Schulz, Martin, II, 27, II, 861
Schürer, Rudolf, I, 1262
Schwerdt, Jörg, I, 683
Seelam, Seetharam R., II, 701
Segal, Michael, I, 633
Sellappa, Sriram, I, 107
Shalit, Daniel, I, 785
Sharma, J.K., II, 1050
Sharov, Dmitri, I, 1087
Sheel, Stephen J., II, 952
Shires, Dale, I, 1199
Shishkova, I.N., I, 502
Shoja, Gholamali C., II, 659
Sibley, G., I, 385, I, 395
Siedel, Edward, I, 11
Sikdar, K., II, 128
Sikorski, Christopher A., I, 1176
Simon, Donald L., II, 1022
Simon, Jens, II, 569
Singer, Bryan, I, 97
Singer, Joshua A., II, 377
Singh, Nirmal, II, 1050
Singh, Vineet, II, 599
Sîrbu, Ioana, I, 911
Skankar, U., II, 1061
Sklower, K., I, 31
Sloot, P.M.A., I, 518, II, 883
Smid, Michiel, I, 683
Smith, Kate A., II, 318, II, 390
Sokolova, N.V., I, 483
Song, S.W., II, 638
Spahr, J.A., I, 31
Spears, J. Brent, I, 1209, II, 531, II, 540
Spinnato, P.F., II, 883
Stankova, Elena N., I, 447
Štědrý, Arnošt, II, 986
Steinberg, Dan, II, 235
Stone, Christopher, I, 1127
Storm, Christian, II, 231
Strand, Gary, I, 149
Strobel, Matthias, I, 708
Stuer, Gunther, I, 423
Sturler, Eric de, II, 108
Sugihara, Kokichi, I, 12, I, 718, I, 728, I, 738

Sundararajan, Elankovan, I, 918
Sundararajan, N., II, 291
Sunderam, Vaidy S., I, 27, I, 263, I, 326, I, 345, I, 367, I, 375, I, 385, I, 395, I, 404
Sung, Andrew H., II, 253
Szmidt, Eulalia, II, 263
Szwarcfiter, J.L., II, 638

Tafti, Danesh K., II, 718
Tam, Vincent, II, 439
Tan, Chih Jeng Kenneth, I, 589, I, 1289
Tanev, Ivan, I, 284
Tao, Jie, II, 861
Tarkov, Michail S., II, 148
Tasso, Sergio, I, 567
Taylor, John, I, 204, I, 212
Teo, Kok Keong, II, 310
Tifenbach, Bradley D., I, 597
Tobis, Michael, I, 175
Tobochnik, Jan, I, 1031
Tomé, José Alberto, II, 217
Tomlin, Alison S., II, 67
Torres, Enrique Alba, II, 689
Tóth, Csaba D., I, 772
Tourancheau, B., I, 233
Trayanov, A.L., II, 1061
Trehel, Michel, I, 294
Tsai, Chang Jiun, II, 429
Tseng, S.S., II, 429
Tudoreanu, Mihail E., II, 791
Turányi, Tamás, II, 67
Turner, Edward L., I, 1137
Tyrakowski, Tomasz, I, 345, I, 367

Udalov, A.A., I, 473
Ueberhuber, Christoph W., I, 243
Unger, Stefan, I, 28
Uozomi, Takashi, I, 284

Vadhiyar, Sathish, I, 41
Van der Vee, Peter, II, 181
Vasupongayya, Sang, II, 943
Veloso, Manuela, I, 97
Vicente, Luís N., II, 95
Vig, Renu, II, 1050
Villard, Laurent, I, 1170
Vincent, J.-M., II, 831
Vinogradov, Alexander, I, 511
Violi, Angela, II, 485
Voelz, Sheri A., I, 212
Völk, Martin, II, 27
Volkert, Jens, II, 781
Volkov, Vladimir, I, 511
Voth, Gregory, I, 1108
Vrooman, Deborah, II, 952
Vuduc, Richard, I, 117

Wang, Hui, II, 245
Wang, Lipo, II, 310
Wanka, Rolf, II, 628
Warnow, Tandy, II, 1012
Wayland, Vince, I, 149
Wen, Qian, II, 701
Westland Chain, James, II, 738
Westrelin, R., I, 233
Wilde, Torsten, II, 871
Willis, Robert J., II, 390
Wilson, Ted, II, 521
Winkler, Manuela, II, 751
Witenberg, A.B., I, 483
Wong, Kwai L., I, 336
Wu, Jenny Xiaodan, II, 447
Wu, Qingxiang, II, 245

Yamada, Yumi, I, 579
Yan, Jerry, II, 771
Yang, Zhengjing, II, 701
Yaşar, Osman, I, 1147, I, 1159
Yelick, Katherine, I, 127
Yeo, Ai Cheo, II, 390
Yoo, Hyeong Seon, I, 928
Yoon, Neville, I, 937
Youn, Hee Yong, II, 912
Youssef, A.M., II, 13

Zanny, Rodger, II, 118
Zatevakhin, M.A., I, 483
Zhao, Wulue, I, 658
Zheng, W., II, 648
Zhong, Wan-Xie, I, 947

Zhu, Jianping, I, 947
Zhu, S., II, 648
Ziegler, Sibylle, II, 27
Zlatev, Zahari, II, 82
Zudilova, Elena V., II, 903