Numerical Simulation of Reactive Flow

Second Edition

ELAINE S. ORAN
JAY P. BORIS

Numerical Simulation of Reactive Flow

SECOND EDITION

Reactive flows encompass a broad range of physical phenomena, interacting over many different time and space scales. Such flows occur in combustion, chemical lasers, the earth's oceans and atmosphere, and stars and interstellar space. Despite the obvious physical differences in these flows, there is a striking similarity in the forms of their descriptive equations. Thus, the considerations and procedures for constructing numerical models of these systems are also similar, and these similarities can be exploited. Moreover, using the latest technology, what were once difficult and expensive computations can now be done on desktop computers.

This book takes account of the explosive growth in computer technology and the greatly increased capacity for solving complex reactive-flow problems that have occurred since the first edition of *Numerical Simulation of Reactive Flow* was published in 1987. It presents algorithms useful for reactive-flow simulations, describes trade-offs involved in their use, and gives guidance for building and using models of complex reactive flows. The text covers both new topics and significant changes in the treatment of radiation transport, coupling, grids and numerical representations, and turbulence. Chapters are arranged in three broad sections: an introductory short course on modeling and numerical simulation; advanced topics in numerical simulation of reactive-flow processes; and, finally, simulations of complex reactive flows.

This new edition is an indispensable guide to how to construct, use, and interpret numerical simulations of reactive flows. It will be welcomed by advanced undergraduate and graduate students, as well as a wide range of researchers and practitioners in engineering, physics, and chemistry.

Dr. Elaine S. Oran is Senior Scientist for Reactive Flow Physics, Laboratory for Computational Physics and Fluid Dynamics, Naval Research Laboratory.

Dr. Jay P. Boris is Chief Scientist, Laboratory for Computational Physics and Fluid Dynamics, Naval Research Laboratory.

Numerical Simulation of Reactive Flow

SECOND EDITION

ELAINE S. ORAN

Naval Research Laboratory, Washington, D.C.

JAY P. BORIS

Naval Research Laboratory, Washington, D.C.

CAMBRIDGE
UNIVERSITY PRESS

CAMBRIDGE UNIVERSITY PRESS
Cambridge, New York, Melbourne, Madrid, Cape Town, Singapore, São Paulo

Cambridge University Press
The Edinburgh Building, Cambridge CB2 2RU, UK

Published in the United States of America by Cambridge University Press, New York

www.cambridge.org
Information on this title: www.cambridge.org/9780521581752

First edition © 1987 Elsevier Science Publishing
Second edition © 2001 Naval Research Laboratory

This publication is in copyright. Subject to statutory exception
and to the provisions of relevant collective licensing agreements,
no reproduction of any part may take place without
the written permission of Cambridge University Press.

First published 1987
Second edition 2001
This digitally printed first paperback version 2005

A catalogue record for this publication is available from the British Library

Library of Congress Cataloguing in Publication data

Oran, Elaine S.

Numerical simulation of reactive flow / Elaine S. Oran, Jay P. Boris. – 2nd ed.

p. cm.

Includes bibliographical references.

ISBN 0-521-58175-3 (hardback)

1. Fluid dynamics – Mathematical models. 2. Chemical reaction, Rate of – Mathematical models. 3. Transport theory – Mathematical models. I. Boris, Jay P. II. Title.

QA911 .O66 2000
532′.05′015118 – dc21

99-059887

ISBN-13 978-0-521-58175-2 hardback
ISBN-10 0-521-58175-3 hardback

ISBN-13 978-0-521-02236-1 paperback
ISBN-10 0-521-02236-3 paperback

To Daniel, Elizabeth, Paul, David, Linda, Karen,
Katrina, Stephanie, Maria, Robert,
and Beauregard

Contents

Prologue *page* xv

1 An Overview of Numerical Simulation 1

1–1. Some Comments on Terminology 1
1–2. A Classification of Models 3
1–3. Statistical Averaging or Chunking 4
1–4. The Hierarchy of Levels of Simulation 5
1–5. The Growth of Computational Capability 8
1–6. The Tyranny of Numbers 10
1–7. A Bridge between Theory and Experiment 11
1–8. Themes of This Book 13
References 14

2 The Reactive-Flow Modeling Problem 15

2–1. Reactive-Flow Conservation Equations 15
2–1.1. Time-Dependent Conservation Equations 15
2–1.2. Physical Phenomena Represented in the Equations 19
2–1.3. Boundary and Initial Conditions 21
2–1.4. Equations of State 21
2–2. Physical Processes in the Conservation Equations 23
2–2.1. Convective Transport 23
2–2.2. Reaction Kinetics 25
2–2.3. Diffusive Transport 26
2–2.4. Radiation Transport 27
2–2.5. Wavelike Behavior 28
2–3. Feedback and Interaction Mechanisms 29
2–3.1. Dimensionless Numbers 29
2–3.2. Specific Interactions 32
2–4. Complications of Multiphase Flow 34
2–4.1. Interactions among Phases 34

2–4.2. An Example of Multiphase Flow Equations 35
2–4.3. The Complexity of Multiphase Models 36
2–5. Finding Input Data 37
References 38

3 Models and Simulation 40

3–1. Developing a Numerical Simulation Model 41
3–1.1. Posing Scientific Problems for Modeling 41
3–1.2. Selecting the Computer System to Use 42
3–1.3. Choosing the Computational Representation 44
3–2. Limiting Factors and Constraints 45
3–2.1. Limitations in Accuracy 46
3–2.2. The Costs of Complexity 48
3–2.3. Using Phenomenological Models 50
3–3. Constructing and Testing the Model 52
3–3.1. Trade-Offs in Selecting and Optimizing Models 52
3–3.2. Steps in Constructing a Simulation Model 53
3–3.3. Testing Programs and Models 55
3–4. Using Time Efficiently 58
3–4.1. Partitioning Time among Tasks 58
3–4.2. Documenting and Retrieving Results of Simulations 59
3–4.3. Some General Programming Guidelines 60
3–4.4. Consider Starting Over 64
3–5. Programming for Parallel Computers 64
3–5.1. Parallel Processing: The Future of Large-Scale Scientific Computing? 64
3–5.2. Programming for Parallel Computers 65
References 68

4 Some General Numerical Considerations 70

4–1. Introduction to Finite Differences 71
4–1.1. Discretizing Time and Space 71
4–1.2. Advancing Equations in Time and Space 73
4–1.3. Qualitative Properties of Numerical Algorithms 75
4–1.4. Approaches to Analyzing Accuracy 77
4–1.5. Types of Finite-Difference Errors 79
4–2. Local Processes and Chemical Kinetics 82
4–2.1. Explicit and Implicit Solutions 82
4–2.2. Asymptotic Solutions 86
4–3. Diffusive Transport 88
4–3.1. The Diffusion Equation 88
4–3.2. An Analytic Solution: Decay of a Periodic Function 89
4–3.3. Explicit and Implicit Solutions 90
4–3.4. The Cost of Implicit Methods 93
4–3.5. Physical Diffusion and Numerical Diffusion 93

4–4. Convective Transport 94
4–4.1. Fluid Dynamic Flows: Advection and Compression 94
4–4.2. The Square-Wave Test Problem 96
4–4.3. Explicit and Implicit Solutions 97
4–4.4. Phase and Amplitude Errors for Common Algorithms 101
4–5. Waves and Oscillations 104
4–5.1. Waves from Coupling Continuity Equations 104
4–5.2. An Explicit Staggered Leapfrog Algorithm 105
4–5.3. A Reversible Implicit Algorithm 108
4–5.4. Reversibility, Stability, and Accuracy 109
4–6. Approaches to Coupling the Terms 110
References 112

5 Ordinary Differential Equations: Reaction Mechanisms and Other Local Phenomena 114

5–1. Definitions and Properties 115
5–1.1. The Initial Value Problem 115
5–1.2. Accuracy, Convergence, and Stability 116
5–1.3. Stiff Ordinary Differential Equations (ODEs) 120
5–2. Overview of Classical Methods 122
5–2.1. One-Step Methods 122
5–2.2. Linear Multistep Methods 123
5–2.3. Extrapolation Methods 126
5–2.4. Leapfrog Integration Methods 127
5–3. Some Approaches to Solving Stiff Equations 129
5–3.1. Stability and Stiff ODEs. 129
5–3.2. Implicit Methods for Stiff ODEs 131
5–3.3. Useful Methods for Solving of Stiff ODEs 132
5–3.4. Summary: Some Important Issues 138
5–4. Useful Information and Techniques 139
5–4.1. Solving Temperature Equations 140
5–4.2. Sensitivity Analysis 141
5–4.3. Integration Packages for Solving ODEs 145
5–4.4. Reaction Kinetics and Thermodynamic Input Data 149
5–5. Reduced Reaction Mechanisms for Reactive Flows 149
5–5.1. The Induction Parameter Model 150
5–5.2. Computational Singular Perturbations 151
5–5.3. Intrinsic Low-Dimensional Manifolds 153
References 154

6 Representations, Resolution, and Grids 159

6–1. Representations of Convective Flows 159
6–1.1. Why Convection Is Difficult to Simulate 160
6–1.2. The Continuity Equation 161
6–1.3. Lagrangian versus Eulerian Representations 162

6–2. Discretizing a Continuous Variable 163
6–2.1. Grid-Based Representations 164
6–2.2. Eulerian Grid Representations 165
6–2.3. Expansion and Spectral Representations 168
6–2.4. Lagrangian and Unstructured Grid Representations 170
6–2.5. Macroparticle Representations 171
6–3. Gridding to Represent Complex Geometry 173
6–3.1. Eulerian, Body-Fitted Grids 174
6–3.2. Overset Grids 176
6–3.3. Unstructured Grids 178
6–3.4. Globally Structured Cartesian Grids 180
6–4. Techniques for Adaptive Gridding 183
6–4.1. Strategies for Refining and Coarsening a Grid 183
6–4.2. Adaptive Mesh Redistribution on Multidimensional Cartesian Grids 187
6–4.3. Adaptive Mesh Refinement on Multidimensional Cartesian Grids 189
6–4.4. AMR and Overset Grids 192
6–4.5. AMR and Unstructured Grids 194
6–5. Resolution and Accuracy on a Grid 196
6–5.1. Errors Introduced by Grid Representations 196
6–5.2. The Influence of Local Errors 198
6–5.3. The Connection between Resolution and Accuracy 198
References 199

7 Diffusive Transport Processes 204

7–1. The Diffusion Equation 205
7–1.1. The Physical Origin of Diffusive Effects 205
7–1.2. The δ-Function Test Problem 206
7–1.3. Short- and Long-Wavelength Breakdowns 210
7–1.4. Numerical Diffusion and Instability 211
7–2. Integration of the Diffusion Equation 211
7–2.1. A Finite-Difference Formula in One Dimension 211
7–2.2. Implicit Solutions Using Tridiagonal Matrix Inversions 212
7–2.3. Large Variations in Δx and D 213
7–3. Nonlinear Effects and Strong Diffusion 215
7–3.1. Nonlinear Diffusion Effects 215
7–3.2. Numerical Techniques for Nonlinear Diffusion 218
7–3.3. Asymptotic Methods for Fast Diffusion 220
7–3.4. Flux Limiters 220
7–4. Multidimensional Algorithms 221
7–4.1. Implicit, Explicit, and Centered Algorithms 221
7–4.2. ADI and Split-Direction Methods 224

7–5. Solving for Species Diffusion Velocities 224
7–5.1. The Fickian Diffusion Approximation 224
7–5.2. An Efficient Iterative Algorithm 226
7–6. Evaluating Diffusive Transport Coefficients 228
7–6.1. Thermal Conductivity 228
7–6.2. Ordinary (or Molecular or Binary) Diffusion 229
7–6.3. Thermal Diffusion 230
7–6.4. Viscosity 230
7–6.5. Input Data for Diffusive Transport Coefficients 231
References 231

8 Computational Fluid Dynamics: Continuity Equations 233

8–1. More on Finite-Difference Methods for Convection 235
8–1.1. The Basic One-Step Method 235
8–1.2. Two-Step Richtmyer Methods 237
8–1.3. The MacCormack Method 238
8–1.4. Padé or Compact Finite-Difference Methods 239
8–1.5. Artificial Diffusion 240
8–2. Positivity, Monotonicity, and Accuracy 240
8–3. Monotone Convection Algorithms and Flux Limiting 246
8–3.1. The Basic Idea of Flux-Corrected Transport 247
8–3.2. Generalizations and Extensions of FCT Algorithms 249
8–3.3. The Effects of Order on FCT 251
8–3.4. Other Approaches to Monotonicity 254
8–4. Coupled Sets of Continuity Equations 256
8–4.1. Lax-Wendroff Methods 256
8–4.2. FCT for Coupled Continuity Equations 258
8–4.3. Multidimensions through Timestep Splitting 260
8–4.4. Considerations for Multidimensional Monotone Algorithms 263
8–4.5. Flux-Corrected Transport Packages 267
8–5. Expansion Methods 270
8–5.1. Galerkin, Tau, and Collocation Approximations 270
8–5.2. Spectral Methods 272
8–5.3. Finite-Element Methods 275
8–5.4. Spectral-Element Methods 277
8–5.5. Wavelet Methods 278
8–6. Resolution and Reynolds-Number Limitations 279
References 281

9 Computational Fluid Dynamics: Using More Flow Physics 286

9–1. Building Flow Physics into CFD Algorithms 287
9–1.1. Equations of Compressible and Incompressible Flow 287
9–1.2. Characteristic Trajectories 290

9–1.3. Time Integration 291
9–1.4. Calculating Fast and Slow Flows 292
9–2. Methods for Fast Flows 292
9–2.1. The Riemann Problem and Godunov Methods 295
9–2.2. The Boltzmann Gas-Kinetic and Flux-Vector Splitting Methods 297
9–2.3. Comparisons and Caveats 299
9–2.4. Shock Fitting and the Method of Characteristics 303
9–3. Methods for Incompressible Flows 305
9–3.1. Pseudocompressibility Methods 307
9–3.2. Projection Methods 308
9–3.3. Implicit Pressure Formulations 310
9–3.4. A Method for Incompressible Flows 311
9–4. Methods for Slow Flows 311
9–4.1. The Slow-Flow Methods 312
9–4.2. An Implicit Flux-Corrected Transport Algorithm 314
9–4.3. Several Implicit Algorithms 318
9–5. Lagrangian Fluid Dynamics 320
9–5.1. Advantages and Limitations of Lagrangian Methods 321
9–5.2. A One-Dimensional Implicit Lagrangian Algorithm 323
9–5.3. Multidimensional Lagrangian Algorithms 330
9–5.4. Free-Lagrange Methods 331
9–6. Lagrangian Particle and Quasiparticle Methods 334
9–6.1. Quasiparticle Methods 335
9–6.2. Smooth-Particle Hydrodynamics 338
9–6.3. Lattice-Gas (Cellular) Automata 340
9–7. Vortex and Contour Dynamics 342
9–7.1. Vortex Dynamics 342
9–7.2. Vortex-in-Cell Methods 344
9–7.3. Contour Dynamics 344
9–7.4. Some Comments on Vortex Methods 345
References 346

10 Boundaries, Interfaces, and Implicit Algorithms 353

10–1. Boundary Conditions 354
10–1.1. General Comments on Boundary Conditions 354
10–1.2. Boundary Conditions for Confined Domains 356
10–1.3. Boundary Conditions for Unconfined Domains 362
10–1.4. Conditions for a Thermal Boundary Layer 366
10–2. Boundary Conditions for High-Order Algorithms 369
10–2.1. Use of Characteristics 371
10–2.2. Problems with Acoustic Radiation in CFD 374
10–3. Interfaces and Discontinuities 375
10–3.1. Resolving Active Interfaces 376
10–3.2. Interface Tracking with Moving Grids 381

10–3.3. Surface-Tracking Methods 381
10–3.4. Volume-Tracking Methods 384
10–3.5. Some Concluding Remarks 387
10–4. Matrix Algebra for Finite Differences 388
10–4.1. Statement of the Problem 388
10–4.2. Exact Solution of the Matrix Equation 390
10–4.3. Tridiagonal Matrices 391
10–4.4. Direct Solutions of the Poisson Equation 393
10–4.5. Iterative Solutions of Elliptic Equations 395
10–4.6. Multigrid Iteration Methods 399
10–4.7. Summary 400
References 401

11 Coupling Models of Reactive-Flow Processes 405

11–1. Standard Approaches to Coupling Multiple Time Scales 406
11–1.1. Global-Implicit Coupling 406
11–1.2. Timestep Splitting 408
11–1.3. Trade-Offs in the Choice of Methods 409
11–2. Coupling Physical Processes in Reactive Flows 411
11–2.1. Coupling for Fluid Dynamics, Species Reactions, and Diffusion 412
11–2.2. Timestep Control 418
11–3. More on Coupling: Warnings and Extensions 420
11–3.1. General Comments 421
11–3.2. Including Gravity 425
11–3.3. Implicit, Lagrangian, and Navier-Stokes Fluid Dynamics 426
11–3.4. Multiphase Flows 428
11–4. Coupling Multiple Stiff Processes 431
11–4.1. The General Formulation 431
11–4.2. An Example 433
11-5. More-Complex Spatial and Temporal Coupling 436
11–5.1. A Classification of Time and Space Scales 437
11–5.2. Intermittent Embedding: An Advanced Coupling Problem 439
References 441

12 Turbulent Reactive Flows 443

12–1. Concepts for Modeling Nonreactive Turbulence 444
12–1.1. An Overview of Current Concepts and Approaches 444
12–1.2. Direct Numerical Simulation 447
12–1.3. Turbulence-Averaged Navier-Stokes Equations 448
12–1.4. Turbulence Modeling and Reynolds-Stress Models 451
12–1.5. Probability Distribution-Function Methods 452
12–2. Large-Eddy Simulation and Subgrid Models 453
12–2.1. Overview of Large-Eddy Simulation 454

12–2.2. The Ideal Subgrid Turbulence Model 457
12–2.3. Four Fortunate Circumstances for Turbulence Simulations 459
12–3. Turbulent Reactive Flows 462
12–3.1. Some Problems in Modeling Turbulent Reactive Flows 462
12–3.2. Overview of Turbulent Combustion 466
12–3.3. DNS and LES of Turbulent Reactive Flows 468
12–3.4. Turbulent Combustion Models 470
12–4. Monotone-Integrated Large-Eddy Simulation 473
12–4.1. Why MILES Should Work 474
12–4.2. Tests of the Underlying Concepts 476
12–4.3. Empirical Evidence that MILES Works for Complex Flows 478
12–4.4. Conclusions and Speculations 480
References 483

13 Radiation Transport and Reactive Flows 488

13–1. The Physical Basis of Radiation Transport 489
13–2. The Equations of Radiation Transport 492
13–2.1. The Radiative-Transfer Equation 492
13–2.2. The Radiant Energy Flux 496
13–2.3. Important Limiting Cases 497
13–2.4. Monte Carlo Methods 498
13–3. Radiation Diffusion Approximations 499
13–3.1. The Standard Radiation Diffusion Approximation 500
13–3.2. The Variable Eddington Approximation 500
13–4. Other Solution Methods 503
13–4.1. The Zonal Method 504
13–4.2. Flux or Expansion Methods 507
13–4.3. The Discrete-Ordinate or S_n Method 509
13–5. Using These Models for Reactive Flows 512
References 518

Index 521

Prologue

Reactive flows encompass a very broad range of phenomena, including flames, detonations, chemical lasers, the earth's atmosphere, stars and supernovae, and perhaps even the elementary particle interactions in the very early stages of the universe. Despite the obvious physical differences among these flows, there is a striking similarity in the forms of the descriptive equations. Thus the considerations and procedures for constructing numerical models of these systems are also similar.

There has been an enormous growth in computational capabilities and resources since the first edition of this book appeared in 1987. What were difficult, expensive computations can now be done on desktop computers. Available hardware has improved almost beyond recognition. Supporting software is available for graphics and for handling large amounts of output data. New paradigms, such as parallel and massively parallel computing using distributed or shared memory, have been developed to the point where they are available to most users. Recipes also exist to interconnect desktop computers to build personal parallel computers.

With the explosive growth in available computer technology, there has been concomitant growth in the use of this technology to solve complex reactive-flow problems having numerous physical processes interacting simultaneously on many different time and space scales. The ability to solve these problems is underpinned by significant developments in numerical algorithms for solving the governing equations. With so many practitioners, many new avenues have been explored, and a number have been developed significantly. Some old ideas have been discarded as not general or accurate enough, and some old ideas have been reinstated because they work particularly well on the new computing systems.

This book is a manual on how to construct, use, and interpret numerical simulations of reactive flows. It can also be used as a guide to using complex computer programs written and provided by others. Constructing numerical simulations means making decisions at each step. These decisions affect the subsequent form and use of the model. For example, we must decide which algorithms to use, how to couple the algorithms together, and how to organize and interpret the results. Understanding the trade-offs these decisions imply means understanding the limitations and uses of the models being developed. Using programs, or parts of programs, provided by others can save an enormous amount of time,

but this requires caution and some effort to make sure that the numerical methods are appropriate and the results are physically correct.

This is not really a book about the *physics* of reactive flows. That topic has been treated beautifully by other authors. Some excellent references are listed in the bibliography at the end of this prologue. This is also not a book about simplified phenomenological models for complex applications, although it can serve as a guide for constructing and testing components of such models.

The readers of this book should at least have advanced undergraduate or beginning graduate school knowledge of the basic principles of chemistry and fluid dynamics and their potential interactions. Some familiarity with the rudiments of numerical methods is helpful. The chapters can be divided into three groups:

- Introductory Short Course on Modeling and Numerical Simulation – Chapters 1 through 4
- Advanced Topics in Numerical Simulation of Reactive-Flow Processes – Chapters 5 through 10
- Simulating Complex Reactive Flows – Chapters 11 through 13

Readers who have written and debugged simple programs to integrate differential equations should be able to follow most of the material in this book and can use the references for more details. Using the advanced chapters to best advantage requires referencing other texts that delve more deeply into selected topics.

Introductory Short Course: Chapters 1 through 4

Chapters 1 through 4 are the foundation for material presented in the remainder of the book and discuss the concepts and terminology used throughout. Chapter 1 describes modeling and simulation generally. Chapter 2 presents the reactive-flow equations and discusses the class of problems they describe. It also views this class in the general context of numerical simulation to solve the equations. Chapter 3 gives an analysis of the limitations and trade-offs that arise in constructing and using computer models. Trade-offs must be made at every stage of developing and using a simulation, and these are determined as much by the computational and human resources available as by the specific requirements of the problem being tackled. Chapter 4 leads the reader into the numerical analysis and computational physics issues that are important for modeling reactive flows by explaining the most basic finite-difference and finite-volume approaches used to integrate the physical processes in the reactive-flow equations numerically.

Advanced Topics in Numerical Simulation: Chapters 5 through 10

These chapters introduce a number of important topics and show the relations among them. Many of the specific topics introduced in Chapter 4 are discussed in greater depth and at a higher level of sophistication in these later chapters. They consider processes that are nonlinear and multidimensional, require variably spaced grids that may move in time, and involve many coupled equations. Chapter 5 describes the spectrum of algorithms for solving ordinary differential equations as well as specific applications to reactive flows. Chapter 6 describes the types of representations in general use and the spatial

and geometric problems of constructing an appropriate computational grid. Chapter 7 describes solutions of diffusion equations that represent multispecies diffusion, thermal conduction, and radiation transport. Chapter 8 describes basic computational methods for a single continuity equation and for a set of coupled continuity equations. Such problems are the core of computational fluid dynamics (CFD). Chapter 9 continues this discussion by describing how the CFD methods may be extended by including more of the known fluid physics. Chapter 10 discusses the numerical implementation of boundary conditions, algorithms for representing and tracking interfaces in the reactive flow, and solutions of matrix equations needed for implicit solution algorithms.

Simulating Complex Reactive Flows: Chapters 11 through 13

These chapters tie the ideas and themes of the book together. They draw from the introductory material in Chapters 1 through 4 and from the separate advanced topic chapters. Chapter 11 discusses fundamental issues and solutions in coupling models of different physical processes together to construct a reactive-flow simulation model. Chapter 12 discusses selected aspects of modeling nonreactive and reactive turbulent flows. Chapter 13 describes ways that radiation transport may be included in a reactive-flow simulation.

Acknowledgments

Many people have helped us in preparing this second edition.

For their technical comments and critique of extensive portions of the book, we thank Eliza Stefaniw, David Jones, and Arnaud Trouvé. We thank the editors and translators of the Russian version of the first edition (MIR 1990): Vladimir Zimont, Pavel Chushkin, Valeri Golovichev, Yuri Markachev, and Lev Shurshalov, who carefully read the entire text of that book and gave us many useful comments and corrections. For their many technical contributions and detailed comments, we thank John D. Anderson, Jr., K.N.C. Bray, C. Richard DeVore, Vadim Gamezo, John Gardner, Fernando Grinstein, Carolyn Kaplan, Alexei Khokhlov, S.H. (Harvey) Lam, Alexandra Landsberg, Christopher Montgomery, David Mott, Gopal Patnaik, K. Kailasanath, Bram van Leer, J. Craig Wheeler, Theodore Young, and Steven Zalesak. For helpful conversations and advice on technical issues, we thank Joseph Baum, Stephen Bodner, Phillip Colella, Bodan Cybyk, Sally Cheatham, Mark Emery, David Fyfe, Michel Lefebvre, Randall LeVeque, Charles Lind, Rainald Löhner, Nikolaos Nikiforakis, James Ott, Julian Piana, Xavier Saint-Martin-Tillet, William Sandberg, Martin Sichel, Nicolas Tonello, Raymond Viskanta, Patrick Vuillermoz, and Paul Woodward. For their technical contributions that influenced the material presented in this book, we thank Graeme Bird, David Book, Teman Burkes, Norman Chigier, Jill Dahlburg, Janet Ellzey, Marie Flanigan, Martin Fritts, Raafat Guirguis, Walter Jones, Ron Kolbe, Sam Lambrakos, Kenneth Laskey, Paul Libby, Eric Loth, Thuong Nguyen, Simone Odiot, Choong Oh, J. Michael Picone, Michel Peyard, Joel Schnur, Patricia Tatem, Paul Urtiew, Fred Williams, James Weber, and Yvette Weber.

We would like to thank Katerina Adams, Sandra Harris, Kathleen Ivan, and Lisa Williams for their help, support, and tolerance while we wrote this book. For their extensive editorial assistance, we especially thank Lisa Williams. We also thank Robert Scott and Kevin Harris for their continuing efforts to provide and maintain the wide range of electronic and graphics tools we used. For permission and help in adapting their

illustrations, we thank C. Richard DeVore, Carolyn Kaplan, Alexei Khokhlov, Charles Lind, Robert Meakin, Nikolaos Nikiforakis, Ravi Ramamurti, and Robert Siegel. Also, we thank James Hyman, Robert Noh, Martin Fritts, Steven Zalesak, and Robert Gelinas who provided figures that were used in this book and in the previous edition.

For their foresight, support, and many years of encouragement at the Naval Research Laboratory and through the Office of Naval Research, we would like to thank Alan Berman, Timothy Coffey, Jack Brown, Herbert Rabin, Alan Schindler, Homer Carhart, Bhakta Rath, Fred Saalfeld, and Paul Gaffney. Our special thanks go to Bhakta Rath for his encouragement and help during the course of writing this edition. We would also like to thank Gabriel Roy of the Office of Naval Research; Julian Tishkoff of the Air Force Office of Scientific Research; Heather Dussault, Helena Wisniewski, and James Crowley, formerly of the Defense Advanced Projects Agency; and Kay Howell formerly of the U.S. Department of Defense High Performance Computing Modernization Program.

For their perseverance and patience with us, we would like to thank Daniel Oran, Elizabeth Boris, Karen Clark, Paul A. Boris, Doris Luterman Surick, and of course, Beauregard Brown. We thank Jonah Brown, Oscar Buneman, Klaus Hein, Bernard Lewis, Keith Roberts, and Roger Strehlow for their technical help and encouragement over the many years we knew each other.

Finally, we would like to thank Florence Padgett, our editor and cheerleader who continually encouraged us and with whom it was delightful to work throughout the preparation of this second edition.

BIBLIOGRAPHY

This book complements a number of other books on computational physics, numerical analysis, computational fluid dynamics, combustion, fluid dynamics, the physics of transport processes, and other reactive-flow texts. Here we list a few of these, leaving detailed and specific references to the end of each chapter. The first edition of this book is also listed below because some material appears there that does not appear in the second edition. This general bibliography is broken into several categories:

Physics of Combustion and Reactive Flows

Combustion, Flames and Explosions of Gases, B. Lewis and G. von Elbe, Academic Press, San Diego, 1987.

Physics of Shock Waves and High-Temperature Hydrodynamic Phenomena, Ya.B. Zeldovich and Yu.P. Raizer, edited by W.D. Hayes and R.F. Probstein, Academic Press, New York, 1967.

Detonation, W. Fickett and W.C. Davis, University of California Press, Berkeley, 1979.

Flames, A.G. Gaydon and H.G. Wolfhard, Chapman and Hall, London, 1979.

Combustion Fundamentals, R.A. Strehlow, McGraw-Hill, New York, 1984.

Combustion Theory, F.A. Williams, second edition, Addison-Wesley, Redwood City, Ca., 1985.

The Mathematical Theory of Combustion and Explosions, Ya.B. Zeldovich, G.I. Barenblatt, V.B. Librovich, and G.M. Makhviladze, Consultants Bureau, New York, 1985.

Principles of Combustion, K.K. Kuo, Wiley, New York, 1986.

Transport Processes in Chemically Reacting Flow Systems, D.E. Rosner, Butterworths, Boston, 1986.

Combustion, I. Glassman, Academic Press, New York, 1996.

Thermal Radiation Heat Transfer, R. Siegel, and J.R. Howell, Hemisphere Publishing Corporation, New York, 1993.

Turbulent Reacting Flows, P.A. Libby, and F.A. Williams, eds., Academic Press, San Diego, 1994.

Computational Combustion and Reactive Flows

Computer Modeling of Gas Lasers, K. Smith and R.M. Thomson, Plenum, New York, 1978.
Modelling of Furnaces and Combustors, E.E. Khalil, Abacus Press, Kent, England, 1982.
Numerical Simulation of Reactive Flow, 1st edition, E.S. Oran and J.P. Boris, Elsevier, New York, 1987.
Numerical Approaches to Combustion Modeling, E.S. Oran and J.P. Boris, eds., AIAA, Reston, Va., 1991.
Molecular Gas Dynamics and the Direct Simulation of Gas Flows, G.A. Bird, Oxford University Press, Oxford, England, 1994.
Computational Methods for Astrophysical Fluid Flow, R.J. LeVeque, D. Mihalas, E.A. Dorfi, E. Müller, Springer, Berlin, 1998.

Computational Fluid Dynamics

Computational Gasdynamics, C.B. Laney, Cambridge University Press, New York, 1998.
Computational Fluid Dynamics, P.J. Roache, Hermosa Publishers, Albuquerque, N.Mex., 1982.
Numerical Computation of Internal and External Flows, Volume 1: Fundamental of Numerical Discretization, C. Hirsch, Wiley, New York, 1988.
Numerical Computation of Internal and External Flows, Volume 2: Computational Methods for Inviscid and Viscous Flows, C. Hirsch, Wiley, New York, 1990.
Computational Fluid Dynamics, J.D., Anderson, Jr., McGraw-Hill, New York, 1995.
Computational Fluid Mechanics and Heat Transfer, J.C. Tannehill, D.A. Anderson, and R.H. Pletcher, Taylor and Francis, Washington, D.C., 1997.

Computational Physics and Engineering

Difference Methods for Initial-Value Problems, R.D. Richtmyer and K.W. Morton, Interscience, New York, 1967.
Computational Physics, D. Potter, Wiley, New York, 1973.
Computational Techniques in Physics, P.K. MacKeown and D.J. Newman, Adam Hilger, Bristol, England, 1987.
Computational Physics, S.E. Koonin, and D.C. Meredith, Addison-Wesley, Redwood City, Ca., 1990.
Introduction to Computational Physics, M.L. De Jong, Addison-Wesley, Reading, Mass., 1996.
An Introduction to Computer Simulation Methods, H. Gould and J. Tobochnik, 2nd edition, Addison-Wesley, Reading, Mass., 1996.
Computational Methods in Physics and Engineering, S.S.M. Wong, Prentice Hall, Englewood Cliffs, N. J., 1992.
Numerical Methods for Engineers and Scientists, J.D. Hoffman, McGraw-Hill, New York, 1992.
A First Course in Computational Physics, P.L. DeVries, Wiley, New York, 1994.
Numerical Methods for Physics, A.L. Garcia, Prentice Hall, Englewood Cliffs, N.J., 1994.

Numerical Methods

Applied Numerical Methods with Software, S. Nakamura, Prentice Hall, Englewood Cliffs, N.J., 1991.
Computing for Scientists and Engineers, W.J. Thompson, John Wiley & Sons, N. Y., 1992.
Numerical Recipes in Fortran 77: The Art of Scientific Computing, W.H. Press, S.A. Teukolsky, W.T. Vetterling, B.P. Flannery, M. Metcalf, Cambridge, N.Y., 1996.
Numerical Recipes in Fortran 90: The Art of Parallel Scientific Computing, W.H. Press, S.A. Teukolsky, W.T. Vetterling, B.P. Flannery, M. Metcalf, Cambridge, N.Y., 1996.

1

An Overview of Numerical Simulation

Reactive flows include a broad range of phenomena, such as flames, detonations, chemical lasers, the earth's atmosphere, stars and supernovae, and perhaps even the elementary particle interactions in the very early stages of the universe. There are striking physical differences among these flows, even though the general forms of the underlying equations are all quite similar. Therefore, considerations and procedures for constructing numerical models of these systems are also similar. The obvious and major differences are in the scales of the phenomena, the input data, the mathematical approximations that arise in representing different contributing physical processes, and the strength of the coupling among these processes.

For example, in flames and detonations, there is a close coupling among the chemical reactions, subsequent heat release, and the fluid dynamics, so that all of the processes must be considered simultaneously. In the earth's upper atmosphere, which is a weakly ionized plasma in a background neutral wind, the chemical reactions among ionized gases and the fluid dynamics are weakly coupled. These reactions take place in the background provided by the neutral gas motions. The sun's atmosphere is highly ionized, with reactions among photons, electrons, and ionized and neutral atomic species, all in the presence of strong electromagnetic fields. A Type Ia supernova creates the heavier elements in the periodic table through a series of strongly coupled thermonuclear reactions that occur in nuclear flames and detonations. The types of reactions, the major physical processes, and the degree and type of coupling among the processes vary substantially in these systems. Sometimes reactions are essentially decoupled from the fluid flow, sometimes radiation is important, and sometimes diffusive transport effects are important. These very different systems, however, are generally all dynamic and unsteady.

1–1. Some Comments on Terminology

The words *model*, *simulation*, *algorithm*, *numerical*, and *computational* are used repeatedly in this book. Even though these words seem easy enough to understand and use in ordinary conversation, their technical use is confused and imprecise. To different individuals and research communities, these words have different and sometimes overlapping meanings. To attack this technical "Tower of Babel" by proposing rigid definitions would

only add more confusion. Here, instead, we discuss some of the nuances of key terms used in this book, and then try to use the terms consistently.

What is generally meant by modeling is much broader than what is generally meant by simulation. A simulation attempts to imitate the dynamic behavior of a system and to predict the sequence of events that control that behavior. Modeling is generally used in a broader, more static sense: a model can represent a dynamic phenomenon in its entirety without specifying its evolution. A model can be a formula, a quantitative relation, a collection of empirical data, a set of equations, or even its equivalent analog circuit.

The objective of modeling is not necessarily to produce an exact copy of a system, but to reproduce certain salient features of the system. Because approximations are made to derive the model, it is understood that the model is an imperfect representation of the system. The result is that a model is invalid in some regimes and imprecise by some amount almost everywhere. A simulation, then, exercises a model or collection of models for a particular choice of physical parameters, initial conditions, and boundary conditions.

Throughout the literature, however, the use of the terms *modeling* and *simulation* overlap, and they are sometimes used interchangeably. The use of the words modeling and simulation has been further confused recently because several communities have adapted the terms to mean the use of very fast, data-driven models to create a virtual reality for interactive training or planning purposes. To combat the confusion caused by this degraded use of the terms, we usually hear the modified descriptions "detailed simulation," "detailed modeling," "high-fidelity simulation," or "fundamental model" to mean the careful scientific and engineering computations that are the subject of this book.

The terms *numerical* and *computational* are often used interchangeably, although they have somewhat different meanings. Numerical analysis, generally a theoretical subject whose main practical application is to solve problems on computers, is not all computational. Computational work in simulations is not all numerical. For example, a reactive-flow computation may involve interactive systems programming, text manipulation, data management, graphics, and so on. Here we consider models and algorithms as computational when they describe how the problem is broken down, represented, manipulated, and stored in the computer. We use the word *numerical* to describe techniques, methods, and algorithms when they concern the calculation of numbers or the quantitative evaluation of formulas and equations.

An *algorithm* is a solution procedure used to implement a model. It is not a model, and it is not necessarily numerical. Not every model has an algorithm associated with it; there simply might not be an algorithm to implement a model. A numerical simulation does not necessarily solve the equations that make up the mathematical model directly. The words *advance* and *integrate* might be more appropriate within the context of a simulation than the word *solve*. As the general reactive-flow problem varies in time, we solve a set of time-dependent, reactive Navier-Stokes equations that have been converted into a set of discretized algebraic equations in the computational model to advance the overall mathematical model. This algebraic set of equations is derived from the original set of partial differential equations in the mathematical model. Algorithms are substituted for the various mathematical terms, for their interactions, and thus for the underlying physical phenomena.

1–2. A Classification of Models

One useful classification of models distinguishes among three levels: fundamental, phenomenological, and empirical. In this sequence of levels, the models are increasingly coarse and are solved correspondingly faster. Making appropriate choices for the level of models to use can produce very practical composite reactive-flow models for solving difficult problems.

A *fundamental model*, often called a *detailed model*, describes the properties or behavior of a system starting with as many basic physical assumptions, or *first principles*, as possible. The problems that such a model can treat are limited in scope because of the expense and complexity of the calculation. Thus the usual objective is to simulate one isolated type of process or interaction. Such models can, however, be extremely detailed and exact in what they do cover. What should be a fundamental or detailed model in a reactive-flow simulation is problem dependent. It is defined by what is currently possible to do for that problem, or what is actually needed to represent the behavior of the system being simulated.

Detailed models can provide constants or other information for use in more general but less rigorous calculations. For example, quantum mechanical calculations of the potential energy surface of a molecule often provide information on the chemical reaction pathways and reaction rates used in more phenomenological continuum representations of reacting fluids. The continuum model itself may be made up of submodels that themselves may be fundamental, phenomenological, or empirical. An important objective of detailed modeling is to develop computational models with well-understood ranges of validity and accuracy. In principle, the broader the range of validity, the more generally useful the model is and the more expensive it is to use.

Phenomenological models and *empirical models* must be used when the time or space scales of the physical processes are too disparate to resolve consistently in one calculation. Macroscopic, averaged models of the small-scale processes become the phenomenological models that appear in the governing equations. For example, in the detailed models of flames considered later in this book, the chemical rate constants, diffusive transport coefficients, and equations of state cannot be determined simultaneously with the convective flow. These three types of models represent atomic-scale processes that have been averaged over assumed Maxwellian distributions of particles to derive the continuous, fluid description. These quantities are phenomenologies representing processes that occur on a scale too small to be resolved in a detailed way in the fluid computation itself.

Often the exact mathematical forms of terms in the governing equations are not known. Alternatively, these terms, though known, may be too complex to evaluate. In these cases, a simpler, approximate form must be used. This approximate *phenomenological* form is usually motivated by physical considerations and contains input data obtained from fits to experimental data or more fundamental theories or simulations. Often global constraints and conservation conditions are used to derive the phenomenological model. Turbulence models are examples of phenomenologies in which complex interactions are modeled by equations with physically reasonable forms. These models are then calibrated by adjusting coefficients to fit data derived from experiments or from more specialized, detailed simulations. Other examples include an energy release model for a chemical system, and a

parametric model for soot formation. A phenomenological model is based on a formula or algorithm that represents our intuitive, qualitative understanding of a particular physical situation. Thus, it must be calibrated with a more basic theory or an experiment. Over time these phenomenologies often acquire the reputation of a fundamental model without the corresponding reliability or accuracy. The Arrhenius reaction rate formula for chemical reactions is a good example of this.

Empirical models are either direct fits of data to a given mathematical formula or are data used directly in tabular form. Because the data are usually derived from experiments, they include extraneous effects, such as measurement interference or equipment calibration errors. A caveat attending such models is that they can only be used for interpolation, not extrapolation. This is true with respect to both the physical parameter range over which the model is valid and the particular physical environment in which it is used. Chemical rate constants are usually empirical models. Other examples of empirical models are tables of the rate at which smoke diffuses through a chamber, the rate a fire spreads through a building, or the rate of chemical energy release in a turbulent flame.

In this book we are interested in numerical simulations of reactive flows in which fluid dynamics, chemistry, and a variety of diffusive transport processes play important, dynamic, and interacting roles. For example, consider a propagating laminar flame, for which we construct a detailed model that solves the compressible Navier-Stokes equations with chemical reactions. The fluid dynamics may be solved by a very accurate numerical method and can be considered a fundamental model of convective transport. The chemical reaction rate coefficients are either phenomenological models if they are taken from fundamental calculations that have been fit into an analytic form, or empirical models if they are fits to experimental data. The equation of state may be a table compiled from experiments. Diffusion constants could be estimates based on intelligent guesses, or they could come from experiments or theoretical studies. This detailed model of a flame would become more phenomenological and less detailed if the set of chemical rate equations were replaced by a simplified "reduced" set of equations, or if the radiative effects were approximated by a single loss term instead of solving the equations of radiative transport.

1–3. Statistical Averaging or Chunking

The reactive Navier-Stokes equations that we use to describe a flame do not solve for the behavior of individual particles. They assume that the medium is a continuous, macroscopic fluid. This continuum approximation incorporates statistical averaging processes performed over the small-scale particle dynamics that we know are present.

The process in which the small-scale phenomena are averaged to produce a global quantity or interaction law has been called *chunking*, which means that microscopic complexity becomes statistically simplified, or chunked at a macroscopic level (Hofstadter 1979). Chunking reduces the number of degrees of freedom needed to describe the behavior of a system. The chunked behavior of a system follows relatively simple laws that approximate the statistical behavior of the microscopically complicated physical system. For example, complete, detailed knowledge of quarks and chromodynamics is not necessary for understanding many aspects of the composition and dynamics of atoms and nuclei. The

detailed interactions of every individual molecule in a system are not needed to explain a propagating laminar flame.

As we concentrate attention on more macroscopic scales and interactions, further chunking is needed to obtain usable models. Nuclear physics, for all but extremely high-energy collisions, chunks the behavior of subelementary particles to create protons and neutrons. Physical chemistry chunks proton and neutron systems to produce the composite particle, the nucleus. Chunking at the level of atoms and molecules gives us the continuum fluid representation. Hofstadter describes the relations among various levels of chunking:

> Although there is always some "leakage" between the levels of science, so that a chemist cannot afford to ignore lower level physics totally, or a biologist to ignore chemistry totally, there is almost no leakage from one level to a distant level. That is why people can have an intuitive understanding of other people without necessarily understanding the quark model, the structure of the nuclei, the nature of electron orbits, the chemical bond, the structure of proteins, the organelles in a cell, the methods of intercellular communication, the physiology of the various organs within the human body, or the complex interactions among organs. All that a person needs is a chunked model of how the highest level acts; and as we all know, such models are very realistic and successful.

A corollary is that chunked models no longer have the ability to predict exactly.

> In short, in using chunked high level models, we sacrifice determinism for simplicity. . . . A chunked model defines a "space" within which behavior is expected to fall, and specifies probabilities of its falling in different parts of that space.

In a complex reactive flow, it is not possible to resolve all aspects of the microscopic particle interactions using a continuum approximation. It is, however, possible to include some of the macroscopic consequences of atomic and molecular phenomena in a fluid model. Chemical reactions, interspecies molecular diffusion, thermal conductivity, temperature-dependent enthalpies and equations of state are all examples of macroscopic models of microscopic processes that result from chunking the microscopic physics.

1–4. The Hierarchy of Levels of Simulation

In this book we are not concerned with elementary particles, but with a more macroscopic view of reactive media composed of molecules and atoms. The interactions of particles at this level result in the processes we want to simulate. Table 1.1 lists the hierarchy of mathematical models used to describe the behavior of systems involving many particles and interactions. Discussions of the derivations of these various approaches can be found in a number of texts (for example, Hirschfelder, Curtiss, and Bird [1954] and Bird [1994]). At the most basic level, these techniques provide very fundamental solutions of the Schrödinger equation describing accurate quantum mechanical interactions of particles, such as might lead to quantum molecular dynamics methods. More general approximations

Table 1.1. Levels of Models of Manybody Interactions

Equation	Solution Method
Schrödinger's equation	Direct solution, density functional theory, (prescribed interparticle forces)
Newton's law $f = ma$	Molecular dynamics (particle-based, prescribed interparticle forces)
Liouville equation equation for distribution function, $F(\mathbf{x}_i, \mathbf{v}_i, t), \quad i = 1, N_p$	Monte Carlo methods (statistical, particle-based methods)
	Direct simulation Monte Carlo
Boltzmann equation $F(\mathbf{x}, \mathbf{v}, t)$ binary collisions (low density) good for gases	Direct solution
Navier-Stokes equation $\rho(\mathbf{x}, t), \quad \mathbf{u}(\mathbf{x}, t)$ short mean free path	Direct solution: finite differences, finite volumes, spectral methods . . . (continuum flow methods)

replace individual particles by continuum fluid elements, as in the Navier-Stokes equations. In between these extremes, there are several levels of statistical and particle-based methods that can be necessary for accurate treatment of some reactive-flow problems.

Molecular dynamics (whether quantum mechanical or classical) is one of the more fundamental levels of this hierarchy. The interactions among the particles may be represented by as basic an interaction as Newton's law in classical representations, or by extremely complex quantum mechanical potentials. The Liouville equation is the fundamental equation of statistical mechanics that describes the distribution function of a gas as a function of the $6N_p$-dimensional phase space of N_p particles (Hirschfelder et al. 1954; Goldstein 1982). Monte Carlo approaches and the Boltzmann equation are both derived from the Liouville equation. Monte Carlo methods exploit the statistical nature of multiple collisions in physical systems and have often been used to describe microscopic systems. When particle densities are high enough that the system of particles can be considered as a continuum, the Navier-Stokes equations are generally used to describe the fluid. There are also various subsets or reduced forms of the reactive Navier-Stokes equations that are commonly used today. These are models that include averages of the behavior of the small continuum scales (for example, to describe turbulent flows, Chapter 12), averages of chemical reactions (to "reduce" the chemical models, Chapter 5), or averages over radiative transport processes (Chapter 13).

Some of the information in Table 1.1 is recast into quantitative form in Figure 1.1, which shows the physical regimes of validity of different types of reactive-flow representations. The regimes of validity are shown in this figure as functions of two important parameters, a characteristic length scale of interest in the system L, and the mean molecular spacing Δ, expressed as the cube root of the inverse of the fluid density. The vertical axis is L/d, a measure of the size of the region of interest, where d is the molecular diameter. The

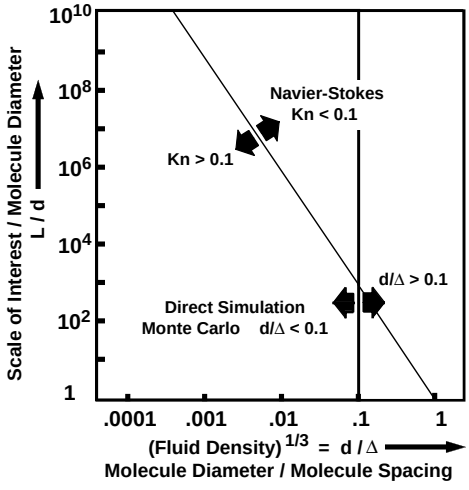

Figure 1.1. Regimes of validity of direct simulation Monte Carlo and Navier-Stokes equations, as a function of the characteristic length scale and mean molecular spacing of a system. In principle, molecular dynamics is valid throughout the entire domain.

horizontal axis shows the dilution of the fluid measured in units of the molecular diameter d divided by the mean spacing of the particles $\Delta \equiv N^{-1/3}$, where N is the number density. As the mean spacing between molecules decreases, the density, $(d/\Delta)^3$, increases. Expressed in this way, the density has a maximum value, at the right side of the figure, of about unity when the molecules are closely packed and the fluid is least dilute.

When the ratio of the mean molecular diameter, d, to the molecular spacing, Δ, is small, so that $d/\Delta \ll 1$ (for example, $d/\Delta < 0.1$), the fluid becomes *dilute*. This regime, which is on the left of the vertical line in the figure, is one in which the statistically based particle method, direct simulation Monte Carlo (DSMC) is an appropriate model (Bird 1994). DSMC is based on the same basic approximations as the Boltzmann equation, but unlike the Boltzmann equation, its regime of validity may be extended to systems where complicated chemistry and three-body collisions occur.

The *Knudsen number*, Kn, is another quantity that is useful in characterizing the regimes of a fluid. This is defined as $Kn \equiv \lambda/L$. Here, L is a characteristic scale length of the system, and λ is the *mean free path*, $\lambda \approx 0.1/Nd^2$, the average distance traveled by a typical molecule between collisions. When λ is small compared to L, and Kn is low enough, the medium behaves as a continuous fluid in which properties such as density, velocity, and energy are well defined at each point. In the collision-dominated regime, where $Kn < 0.1$, the Navier-Stokes model is generally valid.

Because of the different criteria for validity of the Navier-Stokes and DSMC models, both methods can be used in the triangular domain in the upper part of Figure 1.1 where $d/\Delta < 0.1$ and $Kn < 0.1$. When the diagonal line $Kn = 0.1$ is crossed from left to right, the Navier-Stokes equations become valid and are preferred to the DSMC model because of computational efficiency. Thus, this dividing line is usually the practical condition for applicability of DSMC. DSMC is used extensively for high-speed atmospheric reentry problems, where the gas is very dilute and often highly reactive.

Because molecular dynamics is based on a relatively fundamental model, it is valid in principle throughout the entire range of parameters shown. There are no *physical* reasons why it cannot be used for all ranges of densities and system sizes. However, the

computations are very expensive as the number of particles becomes large. In the small triangular regime in the lower right of the figure, neither the Navier-Stokes nor the DSMC model is valid. In this parameter regime, molecular dynamics is, therefore, the preferred model.

This book focuses on finite-volume methods for solving continuum reactive-flow problems. In this approach, time is divided into discrete intervals – *timesteps* – and space into discrete intervals – *computational cells* that define the *spatial grid* or *mesh*. Large sets of discrete variables are defined on the computational cells to approximate fluid variables such as density and velocity. This process puts the equations in a form suitable for numerical computation. We expect the corresponding numerical solutions to converge and become better representations of the continuous fluid variables as the size of the cells and timesteps becomes smaller and smaller. A continuum fluid simulation with cells smaller than a mean free path is subject to problems of physical interpretation even if the model does converge mathematically. Problems that arise due to limits on the attainable resolution are discussed in many parts of this book, but especially in Chapter 6.

Even if practical computational restrictions were removed and arbitrarily good resolution were possible, the continuous variables would still have to be represented in a computer by a discrete set of distinct real numbers with finite numerical precision. It is amusing to note that fluid equations were invented in the first place to simplify the discrete equations of individual particle dynamics. Now we find ourselves reformulating the continuum problem so that we can use digital computers to solve the equations for finite volumes of material. Nevertheless, the continuum set of equations is considerably smaller and simpler to solve than the original equations of particle dynamics.

1–5. The Growth of Computational Capability

Fifteen years ago, when the first edition of this book was written, a two-dimensional computation with 10^6 computational cells was considered very large. Steady-state approximations were the best that could be done for many systems. Today, three-dimensional computations with true dynamics have been performed with more than 10^9 cells. Complex, time-dependent simulations, that were considered an esoteric art confined to a few laboratories with large computers, can now be readily solved by most scientists and engineers on their desktop workstations. All of this reflects continuing exponential advances in computer memory and computational speed.

There have been corresponding changes in our expectations for computing reacting flows. One result of the general increase in computational resources is that proportionately fewer users are trying to develop their own numerical models. They are, instead, using complex, difficult-to-understand simulation programs that were initially written by others. One objective of this book, then, is to explain how to evaluate, use, and interpret the widespread and rapidly growing computational technology available for solving reactive-flow problems.

Figure 1.2 qualitatively illustrates these advances in computer speed and memory as a function of year (based on updating a lecture by Worlton [1988]). The essentially straight line increase of overall performance on the semilog scale means that computer power

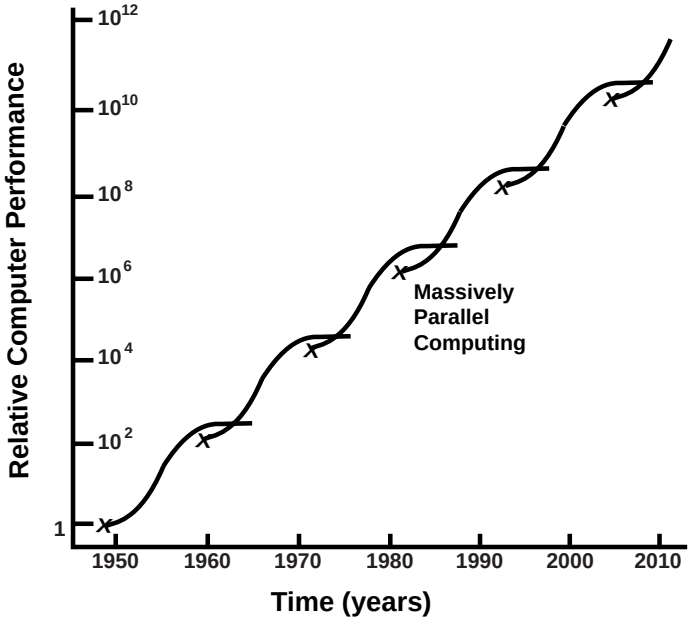

Figure 1.2. Schematic showing the exponential increase of computer power (speed and memory) as a function of time. The crosses indicate the introduction of a major technological breakthrough.

and memory are scaling exponentially. Projections based on current understanding of technology are that this rate can hold for at least another ten years. The figure shows that the exponential performance increase is really composed of a series of piecewise logistic curves, each representing the introduction of a new technology. The result of adding all of the curves is sustained exponential growth. For example, early breakthroughs have included introducing transistors to replace vacuum tubes and semiconductor memory to replace magnetic cores. More recently, we have changed from scalar to vector architectures, and from serial to parallel computing. Such a curve is typical of the growth of an enabling technology, such as high-performance computing. The time-dependent improvement of each component technology is necessarily short term, and saturates at some point. The leveling occurs because the particular technology that allowed the growth becomes too expensive and time-consuming to improve. One obvious feature of this curve is that to continue the growth, and not stagnate, new technologies must regularly be discovered.

Along with the development of high-end computing, there have been the corresponding improvements in personal computers and desktop workstations. As hardware and software become more sophisticated, computing improves across the board. Currently, most parallel supercomputers have about the same speed per processor because they are based on the same or similar chips. The performance increases are driven by microprocessor technology rather than research and development aimed at supercomputer architectures. The current design philosophy tends to use the same basic components (chips) for both supercomputers and powerful workstations. This philosophy may slow the overall pace of computer development, since the high-end systems are less and less the drivers of new

technology. Alternately, it could speed the pace as there is broad-based public demand for faster desktop computing. This is a curious interplay that has practical consequences for scientific computing.

The differences among high-end computers are in their interprocessor communications, memory structures, and architectures. The memory may be globally shared or distributed among the processors. Interprocessor communications may or may not scale linearly as a system expands from a few processors to many processors, and various problems can arise in sending data among processors. At another level, the different architectures cause problems in moving computer programs from one type of machine to another (the portability problem). The current goal is to obtain computational speeds of many teraflops (10^{12} floating-point operations per second), whereas computers delivering many gigaflops (billions of floating-point operations per second) are becoming common.

New advances in algorithms and computer architectures have made a significant contribution to computing reacting flows. Progress has also come through the straightforward application of various existing algorithms as well as from new procedures that combine the standard algorithms with specific features of the new architectures. Future advances could also come from simultaneously combining different kinds of computers, each of which is best suited for a different type of operation, an approach called heterogeneous computing. Another approach uses different sets of processors on the same computer to perform different kinds of operations.

1–6. The Tyranny of Numbers

A detailed numerical simulation model is a program that is executed, or "run," on a computer. The final accuracy and detail we can expect from a numerical simulation of a reactive flow depends on the interplay of a number of factors, including the availability of high-speed computers, specialized numerical algorithms, and fast visualization of the output. The simulations we are concerned with "integrate," or "advance," the approximate time-evolution equations of the model from one time t to another time $t + \Delta t$, where Δt is the *timestep* and is generally smaller than the shortest characteristic times we need to resolve in the calculation. The advancing simulation thus defines the changing state of the system at a sequence of discrete times. The usual assumption is that the state of the system at any time during a timestep can be inferred by interpolating between the state of the system at the beginning and at the end of the timestep.

Typically, hundreds of thousands to millions of numbers are needed to specify the state of a system at any particular time. Consider a detonation propagating in a gas-phase reactive medium. If the problem is two-dimensional, roughly 200×200 discrete points in space are required to give about 1 percent spatial resolution of the flow field and of the chemical species present. If a detailed chemical rate scheme for a diluted hydrogen-oxygen mixture is used, approximately ten species have to be considered in addition to the total fluid density, momentum, and energy density variables. Thus, a minimum of 560,000 numbers are required to specify any state of this modest system at a given time. Realistic simulations are regularly performed for 10^7 variables at a few seconds per timestep, which gives thousands of timesteps per hour of computer time. A major practical problem is how to cope with and effectively interpret so many numbers. We cannot afford

to store all the data from every timestep, nor is it generally useful to do this. Even if the cost of computer memory continues to drop, the time needed to store, compute, and transmit all of these numbers can make a simulation impossibly large and cumbersome.

The tyranny of numbers that plagues most detailed reactive-flow simulations also hampers diagnosis of the computed physical results. To look at the actual numbers associated with just one of the many variables describing a single timestep entails printing out approximately 40,000 numbers, or about forty single-spaced pages of output. If a comparison solution were actually printed out at the same time, eighty pages would be needed. The graphics and diagnostics needed to address this tyranny of numbers effectively can be as time consuming and expensive to develop, apply, and use, as the simulation itself.

In addition to producing large amounts of data, detailed numerical simulations usually require large amounts of computer time. For this reason, simulations are run in segments of generally a few thousand timesteps. Thus, software must be provided to "dump" and save the information describing a particular state of the system at selected timesteps. The calculation can then be restarted from this data at a later time with the option of changing the parameters or the diagnostics.

1–7. A Bridge between Theory and Experiment

Detailed numerical simulation is often the only way to produce general solutions of a complex mathematical model. It is a tool for studying physical systems that bridges theoretical analysis and laboratory experiments. As such, detailed simulations have some of the advantages and the disadvantages of both. This book gives practical procedures for implementing and interpreting chemically reactive-flow models on a computer. It also discusses many numerical algorithms for solving the model equations.

Simulations as Computer Experiments

A detailed computer simulation is not an analytic theory; it does not give equations relating physical variables to each other and to the parameters of the problem. Each simulation is a unique computer experiment that has been performed with one set of geometric, physical, initial, and boundary conditions. The simulation can tell us about something new and unexpected when the model is complete enough, much as a laboratory experiment can teach us something new about the physical environment.

Simulations and experiments contain similar types of errors. In simulations, some of the errors are caused by bugs in the program and are analogous to experimental errors such as leaks or undetected sources of external energy. Calibration errors in an experiment are similar to invalid input constants, parameters, or submodels in a detailed simulation. If the calibrations are incorrect, an experimental apparatus may still be working well. The results, or the interpretation of the data, will be wrong. By the same reasoning, if the chosen values for the controlling chemical rate constants in a simulation are incorrect, the results of the simulation are wrong even though the simulation may be quantitatively accurate for the problem actually specified.

For example, consider an experiment in which streams of hydrogen and oxygen molecules leave a container, mix, and react. Various reactions dominate, depending on the

pressure and density, or on the concentration of impurities or additives present, and so on. To give physically accurate results, a simulation must include a correct chemical mechanism to describe the actual behavior in a natural way. It is usually difficult to know what is missing or what is incorrect when a simulation does not agree with an experiment.

Whereas simulation diagnostics are possible sources of error, the errors associated with experimental diagnostics can also cause problems. Simulation diagnostics can be performed without interfering in the basic computations, but laboratory measurements may perturb the experiment itself. Experimental probes can change the local flow, provide sites for unwanted surface reactions, and absorb heat from the medium. This problem does not exist in the computer experiment.

Both simulations and laboratory experiments benefit greatly from focusing on specific mechanisms and interactions. For example, geometric complications, such as multidimensional effects and wall boundary effects, make both simulations and experiments more difficult. Much can be learned about fundamental interactions by idealizing and simplifying the problem as much as possible. The insight to construct systems for computational study that accomplish this simplification requires a different set of skills from those required for mathematical analysis.

Simulations as Extensions of Theory

Although a simulation may not provide the types of analytic relationships that a theory gives, it provides a similar flexibility. This flexibility is its ability to evaluate the importance of a physical effect by turning the effect on or off, changing its strength, or changing its functional form. This straightforward way of isolating interactions is also an important advantage over experiments. In fact, an experiment is not always a better probe of our physical environment than is a simulation.

Simulations may be used to test the range of validity of theoretical approximations. When a linear theory breaks down, the manner of breakdown can be studied by simulations. Fluid dynamic simulations have been used to study the nonlinear evolution of the Kelvin-Helmholtz and Rayleigh-Taylor instabilities. Theories of transition to turbulence, turbulent boundary layers, and chaotic nonlinear dynamics are other examples where simulations have been used to test and extend theory.

The converse is also true: theory plays a crucial role in validating a numerical model. Exact comparisons with closed-form solutions provide the most valuable benchmarks to verify the computational model. By turning off the terms in the simulation that the theoretical solution does not include, the accuracy and failure modes of parts of the computer model can be evaluated before the program is used to study a more physically complex configuration in which similar errors may go undetected.

Simulation Can Bridge the Gap between Theory and Experiment

Because results obtained from a detailed model may be more comprehensive than those from an analytic theory, detailed models can be used to bridge the gap between theory and experiment. In particular, an effective bootstrap approach to solving problems is to use detailed simulation models to calibrate phenomenological models of specific processes and interactions. These phenomenological models can then be used in the detailed model

to extend its range of validity or its effective resolution. Proposed physical laws can be tested by including the major processes they describe in a simulation and then comparing the results of the simulations and experiments. The use of detailed simulation models to calibrate quantitative understanding of the controlling physical processes is perhaps their most important and fundamental use.

1–8. Themes of This Book

Several interrelated themes recur throughout this book. These themes provide guidelines for developing, carrying out, and interpreting reactive-flow simulations. They present points of view that should be helpful and relate some useful rules of thumb that we have developed after doing many simulations.

Choosing the Type and Level of Simulation

We have already differentiated between modeling based on phenomenologies and empirical laws and modeling from first principles. We also introduced the idea of chunking, which helps determine the level for each component process in the overall simulation. The optimal combination of modeling approaches depends as much on the resources available and the type of answer required as on the parameters of the system being simulated.

Our experience has led us to conclude that reactive flows are generally unsteady, and so the methodology of numerical simulations treated in this book focuses on reproducing time-dependent behavior. We have too often found that steady-state models are unreliable in that they do not reproduce the behavior of the reactive flow: too often they give answers that are both qualitatively as well as quantitatively incorrect. Time-dependent simulations may be used to "march" to a steady state, or their results may be used to produce averages that are suitable for comparing to experimental data.

Building Modular Simulation Models

In a modular approach, the problem is divided into a number of individual physical processes. Models can be built so that each of these processes is calculated accurately and calibrated separately. The resulting algorithms are then coupled together to study the resulting composite reactive-flow mechanisms and interactions. In addition to providing a reasonable organization for attacking the overall problem, this approach allows use of the best algorithm for each aspect of the problem and provides a flexible framework for attacking other problems.

Evaluating Numerical Methods for Accuracy and Efficiency

Throughout this text, we will be evaluating numerical methods with respect to what they can and cannot do in reactive-flow calculations. We try to explain their strong and weak points, emphasizing their accuracy, efficiency, ease of use, and flexibility. We try to introduce more advanced methods in many places and point out some areas to watch for important computational developments.

When the objective is to solve a particular kind of reactive-flow problem, the best methods to use are those which have been extensively tested and debugged. It is crucial

to compare simplified analytic solutions with the output of the simulations when developing numerical simulation models. This check should be done with each process as it is programmed and with combinations of processes whenever possible. It is particularly important to do these tests when developing a new model, and to repeat them after any major changes have been made. Throughout the book, we will return to this theme because success with numerical simulation of reactive flow is as much governed by the reliability of your results as by the results themselves.

Trade-offs

There are trade-offs in every step of the process: in selecting the problem, developing the model, developing the numerical algorithms, building computer programs, and so forth. It is extremely useful to be aware of these and how they affect the solution. Such considerations are a persistent theme from Chapter 3 to the end of the book.

Computational Rules of Thumb

A number of useful "rules of thumb" for simulating reactive flows are given throughout the book. They have evolved from common sense and working through the trade-offs required to find optimum algorithms for each of the relevant physical processes and then for coupling them together.

REFERENCES

Bird, G.A. 1994. *Molecular gas dynamics and the direct simulation of gas flows*. Oxford, England: Oxford University Press.

Goldstein, H. 1982. *Classical mechanics*. 2d ed. Reading, Mass: Addison-Wesley.

Hirschfelder, J.O., C.F. Curtiss, and R.B. Bird. 1954. *Molecular theory of gases and liquids*. New York: Wiley (corrected with notes added, 1964).

Hofstadter, D.R. 1979. *Gödel, Escher, Bach: An eternal golden braid*. New York: Basic Books.

Worlton, J. 1988. *Some patterns of technological change in high-performance computers*. In *Proceedings of supercomputing '88*. Washington, D.C.: IEEE Computing Society Press.

2

The Reactive-Flow Modeling Problem

As described in the previous chapter, the term *reactive flow* applies to a very broad range of physical phenomena. In some cases the equations are not even rigorously known. In this chapter, we first consider the equations of gas-phase reactive flows, which are generally accepted as valid in the continuum regime. This set of time-dependent, coupled, partial differential equations governs the conservation of mass and species density, momentum, and energy. The equations describe the convective motion of the fluid, reactions among the constituent species that may change the molecular composition, and other transport processes such as thermal conduction, molecular diffusion, and radiation transport. Many different situations are described by these equations when they are combined with various initial and boundary conditions. In a later section of this chapter, we discuss interactions among these processes and generalizations of this set of equations to describe multiphase reactive flows.

The material presented in this chapter is somewhat condensed, and is not meant to give an in-depth explanation to those unfamiliar with the individual topics. The purpose is to present the reactive-flow equations, to establish the notation used throughout this book, and then to relate each term in the equations to physical processes important in reactive flows. The chapter can then be used as a reference for the more detailed discussions of numerical methods in subsequent chapters. It would be reasonable to skim this chapter the first time through the book, and then to refer back to it as needed.

2–1. Reactive-Flow Conservation Equations

2–1.1. Time-Dependent Conservation Equations

The equations used to model a gas-phase reactive flow are the time-dependent equations for conservation of the continuum mass density ρ, the individual chemical species number densities, $\{n_i\}$, the momentum density $\rho\mathbf{v}$, and the energy density E. These equations may be written as

$$\frac{\partial \rho}{\partial t} = -\nabla \cdot (\rho \mathbf{v}), \tag{2–1.1}$$

$$\frac{\partial n_i}{\partial t} = -\nabla \cdot (n_i \mathbf{v}) - \nabla \cdot (n_i \mathbf{v}_{di}) + Q_i - L_i n_i, \qquad i = 1, \ldots, N_s, \tag{2–1.2}$$

$$\frac{\partial \rho \mathbf{v}}{\partial t} = -\nabla \cdot (\rho \mathbf{v}\mathbf{v}) - \nabla \cdot \mathbf{P} + \sum_i \rho_i \mathbf{a}_i, \tag{2–1.3}$$

$$\frac{\partial E}{\partial t} = -\nabla \cdot (E\mathbf{v}) - \nabla \cdot (\mathbf{v} \cdot \mathbf{P}) - \nabla \cdot (\mathbf{q} + \mathbf{q}_r) + \mathbf{v} \cdot \sum_i \rho_i \mathbf{a}_i + \sum_i \mathbf{v}_{di} \cdot \rho_i \mathbf{a}_i. \tag{2–1.4}$$

Table 2.1 contains a glossary and dimensional units for the physical variables used in these four equations and in the additional equations given in this chapter.

The first term on the right side of each of equations (2–1.1) to (2–1.4) describes convective fluid dynamic effects. The remaining terms contain the source, sink, coupling, external force, and transport terms that drive the fluid dynamics. The pressure tensor $\mathbf{P}$, heat flux $\mathbf{q}$, total number density N, and energy density E used in these equations are defined by

$$\mathbf{P} \equiv P(N, T)\,\mathbf{I} + \boldsymbol{\tau}, \tag{2–1.5}$$

$$\mathbf{q}(N, T) \equiv -\lambda_m \nabla T + \sum_i \rho_i h_i \mathbf{v}_{di} + P \sum_i K_i^T \mathbf{v}_{di}, \tag{2–1.6}$$

$$N \equiv \sum_i n_i, \tag{2–1.7}$$

and

$$E \equiv \frac{1}{2}\rho \mathbf{v} \cdot \mathbf{v} + \rho \epsilon. \tag{2–1.8}$$

Equation (2–1.5) uses the stress tensor τ defined as

$$\boldsymbol{\tau} \equiv -\mu_m [\nabla \mathbf{v} + (\nabla \mathbf{v})^T] + \left(\frac{2}{3}\mu_m - \kappa\right)(\nabla \cdot \mathbf{v})\mathbf{I}\,. \tag{2–1.9}$$

In addition, expressions are required for the thermal equation of state and the caloric equation of state,

$$P = f_t(N, V, T)$$
$$\epsilon = f_c(N, V, T), \tag{2–1.10}$$

where the generic functions f_t and f_c depend on N, T, and the volume, V. In much of this book we consider gas-phase reactive flows at densities where the thermal equation of state is the ideal gas equation of state,

$$P = N k_B T = \frac{\rho R T}{\overline{m}} = \frac{R T}{\overline{V}}, \tag{2–1.11}$$

where $\overline{m}$ is the mean molecular weight and $\overline{V}$ is the volume per particle (Table 2.1). The caloric equation of state begins with the definitions

$$h_i \equiv h_{io} + \int_{T^o}^{T} c_{pi}\, dT \tag{2–1.12}$$

Table 2.1. Definitions and Units of Variables

Symbol	Definition
$\mathbf{a}_i$	External force per unit mass (cm/g-s^2)
c_s	Speed of sound (cm/s)
c_{pi}	Heat capacity of species i (cm^2/s^2-K)
$\mathcal{C}_i$	Mass-coupling term in energy equation for phase i (erg/cm^3-s)
D_{ik}	Binary diffusion coefficient for species i and k (cm^2/s)
E	Total energy density (erg/cm^3)
$\mathcal{F}_i$	Force term in the multiphase momentum equation for phase i (g/cm^2-s^2)
g	Gravitational acceleration constant (980.67 cm/s^2)
h_i	Specific enthalpy for species i (erg/g)
h_{io}	Specific heat of formation for species i (erg/g) evaluated at temperature T_o
$\mathbf{I}$	Unit tensor (nondimensional)
k^f	Forward chemical rate constant (see Section 2–2.2)
k^r	Reverse chemical rate constant (see Section 2–2.2)
k_B	Boltzmann constant (1.3805×10^{-16} erg/K)
K_i^T	Thermal diffusion coefficient of species i
$\mathcal{L}_i$	Coupling term for multiphase energy equation for phase i (erg/cm^3-s)
m_i	Mass of species i (g)
$\overline{m}$	Mean mass of species in system (g)
M	Mach number, see equation (2–2.12)
$\mathcal{M}$	Mass transfer term in multiphase momentum equation for phase i (g/cm^3-s)
n_i	Number density of species i (cm^{-3})
N	Total number density (cm^{-3})
N_s	Number of chemical species present
P	Scalar pressure (dynes/cm^2)
$\mathbf{P}$	Pressure tensor (dynes/cm^2)
$\mathbf{q}$	Heat flux (erg/cm^2-s)
$\mathbf{q}_r$	Radiative heat flux (erg/cm^2-s)
Q_i	Chemical production rate of species i (cm^{-3}s^{-1})
R	Gas constant (8.3144×10^7 erg/deg-mol)
s	Specific entropy (erg/K)
S	Surface (cm^2)
T	Temperature (K)
$\mathbf{v}$	Fluid velocity (cm/s)
$\mathbf{v}_{di}$	Diffusion velocities of species i (cm/s)
V	Volume (cm^3)
$\overline{V} = 1/N$	Volume/particle (cm^3)
Greek	
α_i	Parameter in multiphase equations, e.g., volume fraction of phase i)
γ	Ratio of specific heats, c_p/c_v
Γ	Circulation, (cm^2/s)
Γ_i	Mass transfer term in multiphase density equation (g/cm^3-s)
ϵ	Specific internal energy (erg/g)

(*continued*)

Table 2.1 (*continued*)

Symbol	Definition
Greek	
κ	Bulk viscosity coefficient (poise, g/cm-s);
λ	Thermal conductivity coefficient (g-cm/K-s^3), also Wavelength
$\Lambda_{i,j}$	Coupling coefficient for external forces for phases i and j (g/cm^3-s)
μ	Coefficient of shear viscosity (poise, g/cm-s)
ν	Frequency (Hz)
ρ	Mass density (g/cm^3)
$\boldsymbol{\tau}$	Viscous stress tensor (dynes/cm^2)
ϕ	Velocity potential (cm^2/s)
φ_i	Parameter in multiphase flow equations, usually volume fraction of phase i
ψ	Stream function (cm^2/s)
ω	Vorticity (s^{-1})
Superscripts	
T	Transpose operation on a matrix
Subscripts	
i, j, k, or l	Individual species or phase
m	Quantity defined for a mixture of species
r	Quantity due to the presence of radiation

which relate the enthalpies $\{h_i\}$ to the heat capacities of the individual reacting species, $\{c_{pi}\}$. The enthalpies are used to define the internal energy,

$$\rho\epsilon = \sum_i \rho_i h_i - P. \tag{2–1.13}$$

The external forces $\{m_i \mathbf{a}_i\}$ in equations (2–1.3) and (2–1.4) can be different for each individual species. For fluid in a gravitational field $\mathbf{g}$, the accelerations are all the same,

$$\mathbf{a}_i = \mathbf{g} \tag{2–1.14}$$

where $\mathbf{g}$ is the acceleration due to gravity. If equation (2–1.4) is rewritten as an equation for the internal energy, $\partial\epsilon/\partial t$, the external force term drops out. The force terms, however, remain in the momentum equation, equation (2–1.3). The additional complications of electric and magnetic forces are neglected here because we assume the gas is un-ionized. When these forces are included, the fluid dynamics equations are not changed greatly, but there are complications and significant additional work from the solution of the associated field equations.

The production terms and loss rates for the different species, $\{Q_i\}$ and $\{L_i\}$ in equations (2–1.2), can be written as

$$Q_i = \sum_j k^f_{i,j} n_j + \sum_{j,k} k^f_{i,jk} n_j n_k + \sum_{j,k,l} k^f_{i,jkl} n_j n_k n_l \tag{2–1.15}$$

and

$$L_i = k_i^r + \sum_j k_{i,j}^r n_j + \sum_{j,k} k_{i,jk}^r n_j n_k. \tag{2–1.16}$$

Here k^f and k^r are forward and reverse reaction rates describing the interactions among the different species denoted by $\{i, j, k, l\}$. These rates may be functions of temperature and pressure.

The set of species diffusion velocities $\{\mathbf{v}_{di}\}$ is found by inverting the matrix equation

$$\sum_k \frac{n_i n_k}{N^2 D_{ik}} (\mathbf{v}_{dk} - \mathbf{v}_{di}) = \mathbf{G}_i, \tag{2–1.17}$$

where the source terms $\{\mathbf{G}_i\}$ are defined as

$$\mathbf{G}_i \equiv \nabla\left(\frac{n_i}{N}\right) - \left(\frac{\rho_i}{\rho} - \frac{n_i}{N}\right)\frac{\nabla P}{P} - K_i^T \frac{\nabla T}{T}. \tag{2–1.18}$$

These diffusion velocities are subject to the constraint

$$\sum_i \rho_i \mathbf{v}_{di} = 0 \tag{2–1.19}$$

which ensures that no net mass flux can arise from the relative interspecies diffusion since the total mass flux is defined as $\rho\mathbf{v}$.

The standard set of equations considered in most fluid dynamics textbooks is a subset of equations (2–1.1) through (2–1.9). Consider a situation in which there is only one species, no chemical reactions, no external forces, and no radiation transport. Then equation (2–1.2) is eliminated, and $v_{di} = 0$, $Q_i = L_i = 0$, $\mathbf{a_i} = 0$ in equations (2–1.3) and (2–1.4), and $\mathbf{q}_r = 0$ in equation (2–1.4). The *Navier-Stokes equation* is the momentum equation, equation (2–1.3), reduced in this way. Further, when κ and μ are zero, the reduced form of the momentum equation is called the *Euler equation*. More commonly the reduced equations for ρ, $\rho\mathbf{v}$, and E are called the *Navier-Stokes equations*. This reduced set with κ and μ set to zero are called the *Euler equations*. We sometimes refer to the complete set of equations (2–1.1) through (2–1.9) as the *reactive-flow Navier-Stokes equations*.

The Navier-Stokes equations consist of two parts: the convective transport terms, which are the first terms on the right-hand side of the equations, and source and diffusive transport terms. The pressure tensor has two parts, one associated with compression and the other associated with viscous dissipation. In general, κ is negligible except in the study of shock-wave structure and absorption and attenuation of acoustic waves. As it is otherwise generally small, it is often dropped from consideration.

2–1.2. Physical Phenomena Represented in the Equations

The complete set of reactive flow equations has been so difficult to solve that entire disciplines and scientific communities flourish solving subsets of these equations for particular applications. For example, hydrodynamics applications eliminate individual species interactions and even the energy equation, and assume the flow is incompressible ($\nabla \cdot \mathbf{v} = 0$).

Aerodynamics problems often deal with time-independent, nonreacting flows over curved surfaces. Chemical kinetics research often considers the chemical reactions without any of the convection and diffusive transport effects. Chemical processing problems often require the solution of a combination of the chemical kinetics and diffusive transport equations. The coupled set of equations (2–1.1) through (2–1.9) also can be used to describe laminar and turbulent flows, and flame and detonation phenomena. Various sets and subsets of the equations and solution techniques described in this book are applicable to these systems and to many other kinds of reactive flows. The methods on which we focus, however, may be used to solve the complete, complicated set of reactive-flow equations.

Inspection of the various terms in the reactive-flow equations shows that there are basically five types of physical processes that can change the species densities and transport energy. Three of these – species reactions, diffusive transport, and radiation transport – originate in the atomic and molecular nature of matter. The other two – convection and wavelike properties – are collective phenomena.

1. *Convection (or convective transport) describes the motion of fluid quantities in space, how fluid parcels interpenetrate, and how compression changes the density.* Convective effects are represented in the equations by fluxes of conserved quantities through volumes; for example, $\nabla \cdot \mathbf{v}X$ where X is ρ, n_i, E, $\rho\mathbf{v}$ or P. Convection is a continuum concept, which assumes that quantities such as density, velocity, and energy are smoothly varying functions of position. The fluid variables are defined as averages over the actual particle distributions, so that only a few degrees of freedom are necessary to describe the local state of the material. The specific atomic and molecular effects in the reaction and diffusion processes described below are deviations from these macroscopic effects.
2. *Reactions among constituent species (e.g., chemical kinetics, atomic physics, or thermonuclear reactions) are represented by the production and loss terms Q_i and $L_i n_i$ in equation (2–1.2).* These are called "local phenomena" because they do not depend on spatial gradients. Other examples of local phenomena are phase changes, external source terms such as laser or spark heating, and sinks such as optically thin radiation loss. The macroscopic form of these effects used in the continuum equations arises from processes that average over microscopic effects. They are only truly local in the continuum limit.
3. *Diffusion effects (or diffusive transport) are represented generally as a divergence of a flux, $\nabla \cdot Y$ where Y may be $n_i \mathbf{v}_{di}$, $\mu\nabla\mathbf{v}$, or $\mathbf{q}$.* The stress tensor terms in equation (2–1.5) describe the diffusive effects of viscosity. The last terms in equation (2–1.6) describe the change in energy due to molecular diffusion and chemical reactions. Note that the species diffusion velocities, $\{\mathbf{v}_{di}\}$ in equation (2–1.2), also appear in the energy equation (2–1.6) representing part of the heat diffusion process. The spatial derivatives of atomic and molecular quantities enter the equations through these terms.
4. *Radiation transport is represented by the term $\nabla \cdot \mathbf{q}_r$ in the energy equations.* Radiation-transport phenomena describe the change in energy and material properties of a system as photons are emitted and absorbed. The $\nabla \cdot \mathbf{q}_r$ form shown in equation (2–1.4) disguises the complexity by representing all of the physics that

occurs as if it were a simple diffusion term. In fact, $\mathbf{q}_r$ is often defined as a sum of terms found by solving a complicated integro-differential equation that considers complicated wavelength and angular dependencies and summarizes microscopic material processes. Finding $\nabla \cdot \mathbf{q}_r$ is the subject of Chapter 13.

5. *Wavelike and oscillatory behavior are described indirectly in the reactive flow equations by coupled sets of continuity equations.* In wave phenomena, energy is not carried across a system by extensive fluid or particle motions. It is transferred from one element of the fluid to others by waves, which may travel much faster than the fluid velocity. The main types of waves considered are shock waves, which move as discontinuities through the system, and the closely related sound waves, in which there are alternating compressions and rarefactions in density and pressure of the fluid. Another type of wave that can occur in these equations is a gravity wave.

2–1.3. Boundary and Initial Conditions

A specific solution of the reactive-flow equations is determined by the initial conditions, the set of reacting species present and their fluid dynamic, thermophysical and chemical properties, and the boundary conditions that describe the geometry of the system and the exchange of mass, momentum, and energy occurring between the system and the rest of the physical world. Boundary and initial conditions select out the solution that applies to the particular problem under study from the many families of possible solutions. The physical and numerical implications of the particular initial and boundary conditions specified also influence the solution techniques used.

It is often difficult to specify boundary conditions that are simple, mathematically tractable, and also physically accurate and stable. Inflow, outflow, free surfaces, fluid interfaces, and flows about solid objects all require different analytic representations and the corresponding numerical models are different. Sometimes imposing physically accurate boundary conditions rules out certain solution methods. We return to this discussion on boundary conditions in Chapter 10.

Specifying the initial conditions should be easier than specifying boundary conditions because initial conditions are input at the beginning of the calculation, rather than appended as constraints on the calculation at every timestep. In practice, however, specifying all the necessary chemical kinetic, equation of state, molecular, and thermal transport data for the particular materials in the system can be an extremely time-consuming process and may introduce large errors.

2–1.4. Equations of State

An equation of state is a relation at thermodynamic equilibrium among the variables P, V, N, and T. Having correct, sufficiently accurate equations of state is an integral part of thermodynamics, fluid dynamics, and chemistry. For example, equations (2–1.1) through (2–1.4) are a set of four equations for the variables ρ, $\rho\mathbf{v}$, E, and $\{n_i\}$, but there is another unknown, P. If we assume a perfect gas, we can add equation (2–1.11), but now there is another unknown, T. The equation needed to close the system is the caloric

equation of state, equation (2–1.12). Thus, in general descriptions of fluid dynamics and reactive flow, two equations of state, of the generic form of equation (2–1.10), are needed.

The ideal gas equation of state, equation (2–1.11), works best for a dilute gas. While it is usually used for most gases even at atmospheric pressures and densities, collections of $P - V - T$ data for specific systems show that it begins to break down even for gases under atmospheric conditions. More complex equations of state consist of expansions of $P\overline{V}/RT$, where the expansion coefficients are determined either theoretically or by fitting experimental data.

Such expansions include the Van der Waals equation of state,

$$\left(P + \frac{a}{\overline{V}^2}\right)(\overline{V} - b) = RT \tag{2–1.20}$$

where a results from the attractive molecular forces, and b accounts for the finite volume of the molecules (i.e., for short-range repulsive forces). This equation of state can be fit to $P - V - T$ data for many gases over small ranges of T. For a single equation valid over larger ranges, it is possible to use one of many empirical relations that have adjustable constants. Often, however, these constants cannot be easily related to molecular forces.

In the virial equation of state, $P - V - T$ data are fit to the form

$$\frac{P\overline{V}}{\rho RT} = 1 + \frac{B(T)}{\overline{V}} + \frac{C(T)}{\overline{V}^2} + \frac{D(T)}{\overline{V}^3} + \cdots \tag{2–1.21}$$

This results in equations of state that are valid for larger ranges of $P - V - T$ data than equation (2–1.20). The virial equation of state has a sound theoretical foundation in that each coefficient may be interpreted in terms of molecular properties. The second term represents deviations from the ideal gas due to two-molecule interactions; the third term is due to three-molecule interactions, and so on.

It is possible to derive the equation of state from statistical mechanics using knowledge or assumptions about the forces between the particles. These derivations recover the ideal gas law in one limit and can reproduce the virial expansion to account for deviations. The virial equation of state does not apply when the material is far out of the normal pressure and temperature ranges. This occurs in astrophysics, inertial confinement fusion, nuclear physics, liquids, shocked condensed-phase physics, and many other situations. In these situations, the equation of state might be derived from first principle knowledge of the particle forces, an equation with constants fit to data, or a set of graphs or tables. For example, in astrophysical applications it is necessary to account for the pressure due to radiation and the degeneracy of the electron gas. For shocked condensed-phase material, the Grüneisen equation of state limits the material compression possible. For ionized and partially ionized gases, the Saha equation of state is often used. Excellent general references for gases and liquids are given by Hirschfelder, Curtiss, and Bird (1964) and Eliezer, Ghatak, and Hora (1986). A good review of the Virial equation is given by Mason and Spurling (1969). A review of equations of state for shock wave physics is given in the book by Gathers (1994), and for astrophysical applications in the books by Kippenhahn and Weigert (1990) and Chabrier and Schatzman (1994).

2–2. Physical Processes in the Conservation Equations

This section considers important concepts and quantities relating to the five main processes occurring in the reactive-flow conservation equations and introduces the terminology and notation used throughout the book. In addition, we try to expose the approximations made in formulating these equations and point out the limits of their applicability.

2–2.1. Convective Transport

The term *convection* (or *convective transport*) means the motion of fluid quantities in space, how fluid parcels interpenetrate, and how compression changes the density. In some of the disciplines in which fluid dynamic effects are important, such as atmospheric physics and astrophysics, the term convection has often taken on a much more restricted definition than in the general fluid-dynamics sense used throughout this book. Convection is expressed through the first flux divergence term on the right-hand side of each of the Navier-Stokes equations. Here we review several transformations used to express the equations in forms suitable for different solution methods.

The *vorticity*, $\boldsymbol{\omega}$, is defined by

$$\boldsymbol{\omega} \equiv \nabla \times \mathbf{v}. \tag{2–2.1}$$

Vorticity is a vector quantity describing the rotation of a fluid parcel. It satisfies an evolution equation that can be derived from the density and momentum equations using equation (2–2.1),

$$\frac{\partial \boldsymbol{\omega}}{\partial t} + \boldsymbol{\omega}\nabla \cdot \mathbf{v} = \boldsymbol{\omega} \cdot \nabla \mathbf{v} + \frac{(\nabla \rho \times \nabla \cdot \mathbf{P})}{\rho^2}. \tag{2–2.2}$$

The net *circulation* around a surface area S is defined as the integral of the fluid velocity along the perimeter of that area,

$$\Gamma = \oint_L \mathbf{v} \cdot d\mathbf{r} = \int_{\mathbf{S}} \boldsymbol{\omega} \cdot d\mathbf{S} \tag{2–2.3}$$

where $\mathbf{S}$ is a surface and $\oint_L$ is the line integral around the perimeter of $\mathbf{S}$. The evolution of the net circulation is determined by

$$\frac{d\Gamma}{dt} = \int_{\mathbf{S}} \left[\frac{(\nabla \rho \times \nabla \cdot \mathbf{P})}{\rho^2} \right] \cdot d\mathbf{S}, \tag{2–2.4}$$

an integration of the vorticity source term over the area $\mathbf{S}$. Here $\mathbf{S}$ is assumed to be moving with the local fluid velocity at each point.

When $\boldsymbol{\omega} = 0$, there is no rotation in the fluid. In this case the flow can be described entirely by a *velocity potential*, ϕ, where

$$\mathbf{v} = \nabla \phi. \tag{2–2.5}$$

When the flow described by equation (2–2.2) is also *incompressible*, that is, $\nabla \cdot \mathbf{v} = 0$, then

$$\nabla \cdot \mathbf{v} = \nabla^2 \phi = 0. \tag{2–2.6}$$

This is the Laplace equation for ϕ, and this type of flow is called *potential flow*.

In the general case, the vector velocity flow field $\mathbf{v}(\mathbf{r}, t)$ can be expanded as the sum of three terms,

$$\mathbf{v} = \nabla \times \boldsymbol{\psi} + \nabla\phi + \nabla\phi_p. \tag{2–2.7}$$

The vector *stream function* $\boldsymbol{\psi}$ for the rotational component of the velocity satisfies

$$\nabla \times (\nabla \times \psi) = \boldsymbol{\omega}, \tag{2–2.8}$$

while the compressional component

$$\nabla \cdot \nabla\phi = \nabla \cdot \mathbf{v} \tag{2–2.9}$$

is no longer zero. The third term, $\nabla\phi_p$, is a particular potential solution, which can be added to the rotational and compressional contributions in equation (2–2.6), as both its curl and divergence are zero. This particular solution, usually appearing in theoretical treatments, is chosen to adapt the composite solution for $\mathbf{v}$, given by equation (2–2.7), to prescribed boundary conditions. Chapter 9 has a general discussion of the *div–curl* problem.

The potential-flow component of the velocity field in equation (2–2.7), the part with no curl and no divergence, represents *advection* (or *advective flow*). This component represents translational motion, and is free of both rotation and compression. In one-dimensional problems, advection involves motion at a constant velocity. In multidimensions, the velocity potential arises by solving the Laplace equation, equation (2–2.6), with suitable boundary conditions. There are no sound waves, shocks, vortices, or turbulence in potential flow.

Regimes of fluid flow are usually characterized in terms of the *sound speed*, the velocity at which a small pressure disturbance moves in the fluid. An expression for the sound speed can be derived from the continuity and momentum equations by considering steady motion with no viscous terms. The sound speed, c_s, is

$$c_s = \sqrt{\left.\frac{\partial P}{\partial \rho}\right|_s}, \tag{2–2.10}$$

where the derivative is taken at constant entropy (e.g., see Chapter 8 in Landau and Lifshitz [1959]). For an ideal gas, this reduces to

$$c_s = \sqrt{\frac{\gamma P}{\rho}}. \tag{2–2.11}$$

The local *Mach number* of the flow is defined using equation (2–2.11),

$$M \equiv \frac{|\mathbf{v}|}{c_s}. \tag{2–2.12}$$

Different flow regimes may be characterized by the value of the flow Mach number, as in Table 2.2.

Table 2.2. Mach Numbers Characterizing Flow Regimes

Flow Regime	Mach Range	Comment
Incompressible	$M < 0.4$	Essentially no compression; density is constant moving with flow
Subsonic	$0.4 < M < 1$	Compression becomes important
Transonic	$M \sim 1$	Weak shocks occur
Supersonic	$1 < M \leq 3$	Strong shocks occur
Hypersonic	$M \geq 3$	Very strong shocks occur, most of energy is kinetic energy

2–2.2. Reaction Kinetics

In reaction kinetics, we generally consider unimolecular, bimolecular, and termolecular reactions among the interacting species. These three cases can usually be written as

$$a \overset{k^f, k^r}{\rightleftharpoons} b + c \tag{2–2.13a}$$

$$a + b \overset{k^f, k^r}{\rightleftharpoons} c + d \tag{2–2.13b}$$

$$a + b + m \overset{k^f, k^r}{\rightleftharpoons} c + m \tag{2–2.13c}$$

where here the a, b, c, d, and m indicate different chemical species, and k^f and k^r are forward and reverse reaction rates, respectively.

Equations (2–1.2) are the time-dependent conservation equations for the N_s distinct species number densities. There are usually some additional conservation laws that apply to the systems these reactions represent. For example, in standard combustion applications where the reactions are between atoms and molecules, the total number of atoms of a particular type is constant although the molecular configurations might change. The production and loss rates, $\{Q_i\}$ and $\{L_i\}$ in equations (2–1.2), are not individually conservative as the individual species transmute through specified reactions. The terms in equations (2–1.2) describing changes due to the interactions among species are

$$\frac{\partial n_i}{\partial t} = Q_i - n_i L_i, \qquad i = 1, \ldots, N_s. \tag{2–2.14}$$

Equation (2–2.14) is a set of nonlinear, first-order, ordinary differential equations. They are sometimes called *local equations* because there are no spatial gradients or derivatives. Both the $\{Q_i\}$ and $\{L_i\}$ are linear combinations of products of the number densities $\{n_i\}$ multiplied by reaction rates. Their generic forms are given in equations (2–1.15) and (2–1.16).

The individual forward and reverse reaction rate constants k^f and k^r may be complicated functions of temperature and sometimes pressure. The expressions for the rates commonly encountered in combustion kinetics are usually written in an *Arrhenius* or *modified Arrhenius form*,

$$Ae^{-C/T} \quad \text{or} \quad AT^B e^{-C/T}, \tag{2–2.15}$$

respectively. The validity of these expressions depends on the existence of a Boltzmann distribution in the vibrational and electronic states of the participating molecules. Often, however, equation (2–2.15) is not completely adequate to describe the reactions. More complex forms of reaction rates, or a table of values as a function of temperature and pressure, may be more appropriate. Sometimes the individual vibrational or electronic states must be considered separately. This occurs in certain atmospheric, solar, and chemical laser applications.

When equation (2–2.14) is coupled to the equation of state, equations (2–1.10) through (2–1.13), the temperature can be determined from an algebraic equation expressing energy conservation. When these thermophysical quantities are not known, it is often possible to solve an ordinary differential equation that describes the change in temperature due to chemical reactions. This more complicated situation occurs in atmospheric chemistry problems in which the vibrational or electronic states are not in equilibrium.

When the chemical reactions release or absorb a great deal of energy, the reaction kinetics are strongly coupled to convection through the pressure and density. This is the situation for flames and detonations. When the chemical reactions are approximately thermoneutral – that is, when little energy is absorbed or emitted by the chemical reactions or when the reactants are highly dilute – there is only limited feedback between the fluid dynamics and the chemistry. Then the major effect of chemistry on the fluid dynamics may be through a change in the number density, which causes changes in pressure. The thermoneutral assumption is correct to first order when we relate the ion chemistry of the earth's ionosphere to neutral wind motion, or the density of the heavy ions to the motion of the bulk plasma in the solar atmosphere. When the reactions are highly endothermic or exothermic, as in combustion, or when ionization is significant, as in laser-produced plasmas, the strong coupling between chemistry and fluid dynamics cannot be ignored.

2–2.3. Diffusive Transport

The viscous component of the pressure tensor, P, and heat flux vector, $\mathbf{q}$, represent diffusive fluxes of momentum and energy in a region of space. There are also flux vectors for species diffusion, $\{\rho_i \mathbf{v}_{di}\}$, that are constrained by equation (2–1.19). These fluxes are written in terms of the transport coefficients and gradients of mass, velocity, and temperature. Such relations are valid when conditions deviate only slightly from equilibrium, so the underlying particle distribution functions are nearly Maxwellian and variations of average fluid properties over a mean free path are small. When the gradients are small enough, the exact fluxes are well approximated as first derivatives of density, velocity, or temperature.

Diffusive transport processes express the transfer of mass, momentum, or energy due to the presence of gradients in concentration, velocity, or temperature. The transport coefficients and the gradients with which they are associated are given in Table 2.3. Molecular diffusion represents a diffusive transfer of mass and energy due to a species concentration gradient. Viscosity represents diffusive transfer of momentum due to a velocity gradient. Thermal conduction and thermal diffusion represent transfers of energy due to

Table 2.3. Transport Coefficients and Associated Gradients

Diffusion Coefficient	Symbol	Associated Gradients
Molecular Diffusion	D_{ij}	$\nabla(n_i/N), \nabla T, \nabla P$
Viscosity	μ_m	$\nabla \mathbf{v}$
Thermal conduction	λ_m	$\nabla T, \nabla P$
Thermal diffusion	K_j^T	∇T

a temperature gradient. Extensive descriptions of diffusive transport properties are given in the classic books by Hirschfelder, Curtiss, and Bird (1964) and Chapman and Cowling (1970). Additional descriptions are given by Reid, Prausnitz, and Sherwood (1977), and discussions related to combustion are given by Strehlow (1984) and Kuo (1986).

2–2.4. Radiation Transport

Equation (2–1.4) includes the effects of energy transport by radiation, as expressed in the term $\nabla \cdot \mathbf{q}_r$. This transport process could be either convective or diffusive in nature, and these distinctions are discussed in detail in Chapter 13. In addition to a contribution to the heat flux vector $\mathbf{q}$, there are also contributions from radiation to the pressure tensor and the internal energy density. For gas-phase flames and detonations, these additional contributions are usually small and so are neglected in equations (2–1.3) and (2–1.4). For other applications, such as astrophysical phenomena, the effect of radiation on the pressure can be significant.

In general, the term representing the radiation flux in equation (2–1.4) can be written

$$\nabla \cdot \mathbf{q}_r = R_E - R_A \tag{2–2.16}$$

where R_E is the rate of emission of radiation per unit volume and R_A is the rate of absorption of radiation per unit volume. Usually the absorption rates are proportional to the densities of the absorbing species. Often radiation transport is a strong function of wavelength. This occurs when particular atomic transitions and hence particular spectral lines are important. In these cases the flux has to be treated as an integral over wavelength, λ,

$$\mathbf{q}_r(\mathbf{r}) = \int_0^\infty \mathbf{q}_\lambda(\mathbf{r})\, d\lambda. \tag{2–2.17}$$

Generally $\mathbf{q}_\lambda$ is derived in terms of the coefficients of absorption of a material, $\alpha_\nu(T, \rho)$.

There are two cases that can be treated rather directly. The first is the case of an opaque fluid, such as might occur in the hot gases in a car engine. This situation is referred to as *optically thick*. Here the absorption is so intense that the photons travel only a very short distance before being absorbed. The radiation is essentially in local thermodynamic equilibrium with the material. For this application, a mean free path for the photons making up the radiation field can be defined and radiation transport can then be treated as a form of diffusion. The other limit occurs in some relatively nonluminous flames and other low-

density situations when the reacting fluid is essentially transparent to the radiation. There is little self-absorption so that the term R_A in equation (2–2.16) is very small and the emission term, R_E, dominates. This situation is often referred to as the *optically thin* limit and is characterized by the free-streaming convection of photons.

A more detailed explanation of the processes involved and a number of specific methods for treating radiation transport in a numerical model are discussed in Chapter 13. References to the physics of these processes are Zeldovich and Raizer (1966), Rybicki and Lightman (1979), Mihalas and Mihalas (1984), Siegel and Howell (1993), and Modest (1993).

2–2.5. Wavelike Behavior

A *wave* is the term often applied to any front or discontinuity that moves through a system. It is commonly used to refer to flames, detonations, pressure pulses, shocks, and so on. More technically, a wave is associated with transport of energy caused by a back and forth, oscillatory energy transfer between two of the basic processes in the reactive-flow equations.

The four types of processes discussed above are all identified with specific terms in the reactive-flow equations and the solution techniques and algorithms for them will be the subjects of later chapters in the book. Wave phenomena are generated by the interaction between distinct terms in the Navier-Stokes equations and will not be the subject of a separate chapter. Nevertheless, waves are important because they exist in all of the solutions, and they transport energy over long distances without a corresponding motion of the material. Thus energy in the matter at one location can be moved by waves to another location where the chemical species may be different or the matter is in a different state.

A key ingredient of waves is the inertia term, $\partial \rho \mathbf{v}/\partial t$, in the momentum equation. The fluid inertia causes a time delay between the application of a force and the response of the medium to that force. This time delay determines the frequency and the effective speed of propagation of the wave. The existence of a characteristic frequency also allows resonances in which a very small input of energy over a long period of time, if correctly phased with the wave, can lead to very large amplitudes.

Later, in Chapter 9, we will show explicitly from the Navier-Stokes equations how the velocity at which energy is convected can be different from the velocity at which material is convected. It is only a small step from this to see that energy can move a long distance while the material executes only a small oscillation with no net motion. Indeed, the main types of waves – acoustic and gravity – are characterized by these oscillations.

In the case of acoustic waves, the momentum and the energy (or equivalently, the mass density) continuity equations interact. Compression of the fluid in one region increases the pressure in this region. In turn, the spatial gradient of the local pressure accelerates the fluid, which causes a compression nearby at a later time. This propagates the wave. Acoustic waves can become very nonlinear, leading to combustion instabilities, shocks, and even detonations in reactive flows. Even in situations where the compression is small and the sound waves are quite linear, the effects can be enormous. Sound waves from vortex merging in a shear flow, for example, trigger instabilities far upstream of the merging events. This qualitatively changes the nature of the flow.

Gravity waves store energy locally through vertical displacement. For this energy repository to exist, significant density gradients should exist in the undisturbed flow. Here also, inertia provides the delay mechanism that allows a wave to propagate. As with acoustic waves, the gravity-wave speed can be much faster than the motion of the fluid itself. The limiting situation of a gravity wave is the propagation of waves on the free surface of a body of water. As with acoustic waves, strong nonlinearity in gravity waves can cause appreciable local dissipation. Gravity waves even display shock phenomena (hydrodynamic bores) just as sound waves do.

Gravity waves can also propagate without gravity. Any acceleration field will do. A good example is a large rotating vortex with an axisymmetric (radial) density variation. If the density is lower on the inside, gravity waves can propagate around the interior of the vortex. If the high density is on the inside, the vorticity generation from the slightly misaligned density and pressure gradients grows to a nonlinear level exponentially, a phenomenon called *Rayleigh-Taylor instability*. These gravitational instabilities give rise to long-lasting turbulence and mixing that can strongly influence combustion.

2–3. Feedback and Interaction Mechanisms

In reactive flows, the processes described above do not usually occur independently of each other. (Indeed, waves are really the interaction of two processes.) These processes interact in a variety of fundamental ways to control crucial aspects of reactive flows. A discussion of these interactions could fill several volumes.

2–3.1. Dimensionless Numbers

For most reactive flows, it is difficult if not impossible to solve the governing equations analytically. This is particularly true when two or more of the fundamental processes are interacting. When theory cannot provide a solution, and we wish to study general properties of the system, it is often useful to study geometrically and physically similar systems and make comparisons. For example, consider two systems in which the size, fluid conditions, or even the type of fluid may differ. In some cases, it appears that the behavior of similar systems should scale with each other. Methods such as *dimensional analysis* are used when the variables are known but the equations are not. Methods such as *inspection analysis* of the governing equations are used when the equations are known. These methods lead to a host of dimensionless numbers that are ratios of physical quantities describing the system. With care, and an eye on the underlying assumptions involved, systems may be characterized by the values of these dimensionless numbers. A more detailed discussion is given in a number of texts including Brodkey (1967) and Incropera and DeWitt (1990).

As an example, consider the momentum equation, equation (2–1.3). A component or structure in the flow can be characterized by a length scale l in the fluid, a time scale t for the flow, a velocity v, and an external force ρg. Now consider another flow with geometrically similar boundaries, so that this second flow is characterized by parameters indicated by primes: l', t', and v'. If these flows are similar, we can write down a set of relations of the form $l = c_1 l'$, $t = c_2 t'$, and so on, and replace the variables in equation (2–1.3) by the primed variables. Then equating the coefficients in the regular and primed

equations, a number of relations emerge. For example,

$$\frac{lv\rho}{\mu} = \frac{l'v'\rho'}{\mu'} \equiv Re = constant. \tag{2–3.1}$$

Thus the quantity $lv\rho/\mu$ must be the same in both systems if they are to be considered similar. This dimensionless ratio is the *Reynolds number*, the ratio of inertial to viscous forces. We also have

$$\frac{v^2\rho}{P} \equiv \frac{1}{Eu}\left(= \frac{v^2\gamma}{c_s^2}\right). \tag{2–3.2}$$

This relation defines the *Euler number*, the ratio of pressure to inertial forces. The term in parentheses on the right side of equation (2–3.2) is the special case when the gas obeys the ideal gas law. This definition shows that Eu is the gas constant times the square of the Mach number. If the body force is gravity, so that $f = \rho g$, we obtain

$$\frac{v^2}{lg} \equiv Fr, \tag{2–3.3}$$

which defines the *Froude number*. The Froude number is the ratio of kinetic to gravitational potential energy.

The equations of motion for two flows will be the same if their Reynolds numbers, Mach numbers, Froude numbers, and the ratio of their specific heats are equal. Inspection of the continuity equation does not produce any new dimensionless numbers, but inspection of the energy equation does. These include: the *Peclet number* $\mathrm{Pe} = vl\rho c_p/\lambda$, the ratio of heat transported by convection to heat transported by conduction; the *Prandtl number* $\mathrm{Pr} = c_p\mu/\lambda$, the ratio of momentum diffusivity (kinematic viscosity) to thermal conduction; the *Brinkman number* $\mathrm{Br} = \mu v^2/\lambda T$, the ratio of heat produced locally by viscous dissipation to that transported by thermal conduction; the *mass transfer Peclet number* $\mathrm{Pe, s} = lv/D_{ik}$, the ratio of convective to molecular mass transfer; and the *Schmidt number*, $\mathrm{Sc} = \mu/\rho D_{ik}$, the ratio of momentum diffusion to mass diffusion. These and other dimensionless ratios are defined in Table 2.4 that shows some of the more commonly-used dimensionless numbers. There are many others, some of which can be written as ratios of these numbers. A useful rule of thumb, when using an unfamiliar dimensionless number, is to make sure that you understand how it is defined and exactly how it is nondimensionalized.

Characterizing one system or a set of apparently similar systems by a dimensionless number is often useful. For example, the Reynolds number can often be used to estimate whether or not a flow is unstable to a shear instability. Whether a droplet will break up can sometimes be estimated by evaluating the Weber number. But these nondimensional numbers can also be misleading, because a reactive flow is characterized by many such numbers which may be changing in time and space. For example, the Reynolds number evaluated at some location and at some time might be the same for two systems. The flow stability based on the Reynolds number might be wrong because of the effects of time-dependent chemical reactions and subsequent energy release. Even though the Weber number for breakup of a droplet for an idealized case could be one value, the presence of strong shears in the flow could cause droplets of much smaller radius to

Table 2.4. Dimensionless Numbers

Name	Symbol	Definition	Significance
Atwood	A	$\Delta\rho/(\rho_1+\rho_2)$	Density difference/Density sum
Brinkman	Br	$\mu v^2/T\lambda$	Heat from viscous dissipation/ Heat from thermal conduction
Capillary	Cp	$\mu v/\sigma$	Viscous force/Surface tension force
Crispation	Cr	$\mu\lambda/\sigma l\rho c_p$	Diffusion/ Surface tension
Damköhler	Da_1	$R_i l/v$	Chemical reaction rate/ Convective transport rate
	Da_2	$R_i l^2/D$	Chemical reaction rate/ Diffusive transport rate
	Da_3	$Ql/\rho hv$	Heat from chemistry/ Heat from convection
	Da_4	$Ql/\lambda T$	Heat from chemistry/ Heat from conduction
	Da_5	$Ql/\rho Dh$	Heat from chemistry/ Heat from diffusion
Drag Coefficient	C_D	$(\rho'-\rho)/lg\rho' v^2$	Drag force/Inertial force
Euler	Eu	$P/\rho v^2$	Pressure force/Inertial force
Froude	Fr	v^2/lg	Kinetic energy/ Gravitational energy
Grashof	Gr	$(g\Delta T l^3)/T\mu^3$	Buoyancy forces/Viscous forces
Karlovitz	K	$(l/v)(dv/dy)$	Stretch factor: criterion for area increase in flame front
Knudsen	Kn	δ/l	Convection time/Collision time
Lewis	Le	$\lambda/\rho D_{ij}c_p$ =Sc/Pr	Thermal conduction transport/ Diffusive energy transport (sometimes defined as inverse)
Mach	M	v/c_s	Magnitude of compressibility effects
Newton	Nt	$F/\rho l^2 v^2$	Imposed force/Inertial force
Nusselt	N	$\alpha l/\lambda$	Total heat transfer/ Thermal conduction
Peclet	Pe	$vl\rho c_p/\lambda$	Convective heat transport/ Heat transport by conduction
	Pe,s	lv/D_{ij}	Convective mass transfer/ Molecular mass transfer
Poisseuille	Po	$l\Delta p/\mu v$	Pressure force/Viscous force
Prandtl	Pr	$c_p\mu/\lambda$	Momentum transport/ Thermal conduction
Rayleigh	Ra	$gl^3\beta\Delta T\rho^2/c_p\mu\lambda$	Buoyancy force/Diffusion forces
Reynolds	Re	$lv\rho/\mu$	Inertial forces/Viscous forces
Richardson	Ri	$gl/(\Delta v)^2$	Buoyancy effects/Vertical shear effects
Rossby	Ro	$v/2\Omega l\sin\Lambda$	Inertial Force/Coriolis force
Schmidt	Sc	$\mu/\rho D_{ij}$	Momentum diffusion/ Mass diffusion
Stanton	St	$\alpha/\rho vc_p$	Thermal conduction loss/Heat capacity
Stokes	S	$\mu/\rho\nu l^2$	Viscous damping rate/Vibrational rate
Strouhal	Sr	$\nu l/v$	Vibrational rate/Convective flow rate

(*continued*)

Table 2.4 (*continued*)

Name	Symbol	Definition	Significance
Weber	W	$\rho l v^2/\sigma$	Inertial force/Surface tension forces
Zeldovich	Z	$\alpha'\epsilon_a/(1+\alpha')$	Activation energy/Thermal energy
Legend *			
c_p	Specific heat at constant pressure (cm^2/K-s^2)		
F	Imposed force (g-cm/s^2)		
g	Gravitational constant (cm-s^2)		
l	Scale length, radius (cm)		
Q	$\sum h_{oj} R_i \rho_i$ (g/s^{-3}), where		
R_i	$= P_i/n_i - L_i$, (s^{-1}) see Table 2–1.		
T	Temperature (K)		
v	Characteristic flow velocity (cm/s)		
α	Newton's-law heat coefficient (g/s^{-3}-K), $\lambda(dt/dx) = \alpha\Delta T$		
α'	$Q/(c_p T_o)$, where T_o is unburned-gas temperature		
β	Volumetric expansion coefficient (s^{-1}) where $\beta dT = dV/V$		
δ	Collisional mean free path (cm)		
Δ	Difference between two quantities		
ϵ_a	E_a/kT_b – activation energy/burned-gas temperature		
Λ	Latitude of position on the earth's surface		
μ	Viscosity coefficient (poise, g/cm-s)		
ν	Frequency (Hz)		
Ω	Solid-body rotational angular velocity (s^{-1})		
ρ	Mass density (g-cm^3)		
σ	Surface tension coefficient (g/s^2)		

* Quantities not defined here are defined in Table 2.1

break up. Furthermore, when three or more processes are interacting, a simple ratio can not suffice to characterize the situation. Therefore, nondimensional numbers should be used with caution, though they provide useful guides for characterizing the behavior of a system.

2–3.2. Specific Interactions

Consider the interaction between convection and chemical reactions. Convection causes a volume of fluid to move from one location to another, to penetrate other elements of fluid, and to change the density in response to changing flow conditions along the the path of motion. Each of these convection effects interact differently with ongoing chemical reactions. The nondimensional Damköhler numbers Da_1 and Da_3 relate to these interactions. Because nearby fluid elements remain nearby in potential flow, the main effect of such flows on chemical reactions is to move the reactions with the associated fluid element from one place to another. Although this sounds simple, the numerical problems

of trying to represent continuous fluid motion through a discretized spatial domain have dominated computational fluid dynamics research for decades.

When the vorticity is nonzero, there are gradients transverse to the local direction of fluid motion. These rotational or "shear" motions allow fluid from one place to penetrate and convectively mix with other fluid because these motions can separate two initially close fluid elements. Shear in the local flow also enhances species gradients and causes faster molecular mixing, generally characterized by the mass transfer Peclet number, Pe,s. A natural consequence of large-scale rotation in flows is turbulent mixing, which brings widely separated fluid elements close together where they can mix diffusively and react chemically. Thus vorticity enhances chemical reactivity. Vorticity can also affect chemical reactions in an initially premixed medium by mixing hot reacted fluid with cold, unreacted fluid. Then the temperature rise as well as the introduction of reactive intermediate species increases the rate of chemical reactions.

When the flow is curl free but not divergence free, that is, when the vorticity $\nabla \times \mathbf{v} = 0$ but $\nabla \cdot \mathbf{v} \neq 0$, there are velocity gradients in the direction of motion causing compression and rarefaction. Compression generally accelerates chemical reactions while expansion generally retards them. A runaway effect is possible, in which higher densities and pressures increase the rate of chemical reactions, which in turn increase pressure.

Fluid motions arising directly from local expansions can be violent, but they decrease in strength with distance from the source and stop when the energy release is complete. The indirect effects of expansion are longer lived and can be more important. Once expansion is complete, it leaves pockets of hot reacted gas in the midst of unreacted fuel or oxidizer. These gradients in temperature and density become relatively static and move with the fluid. Embedded density gradients, however, interact with larger-scale pressure gradients or gravity to generate additional vorticity. Vorticity is generated slowly by this passive influence of reaction kinetics on the flow, but the integrated effect of this vorticity can be much larger than even the long-lived vortices generated directly during a rapid expansion. Although these vortices have flow velocities which are only a few percent of the expansion velocities that produced them, they persist for long periods of time after the expansion has stopped. Furthermore, unlike direct expansion, these rotational flows cause mixing.

Each of the five types of processes in equations (2–1.1) through (2–1.19) – convection, chemical kinetics, diffusive transport, radiation, and waves – can interact with each other. So far we have explicitly considered the interactions of convection and energy release from chemical reactions. Perhaps the most important of the remaining interactions are between molecular diffusion and chemical kinetics, governed by the Damköhler number Da_2 and Da_5 (Table 2.4), and between molecular diffusion and convection giving rise to the Peclet numbers. The rotational component of the flow generates vorticity and causes fluid elements to interpenetrate. This interpenetration increases the surface area of the interfaces between different materials and, by shearing the fluid and stretching the vortices, increases the strength of gradients. Thus diffusive transport of all types is enhanced by direct interaction with the rotational component of fluid flow. The strongest effects of molecular diffusion on the convection occur indirectly through effects such as heat release resulting from mixing fuel and oxidizer. The direct interactions between molecular

diffusion and chemical kinetics occur because each process changes the relative abundance of the species present in a fluid element. Molecular diffusion does this through moving new material into the volume, and chemical kinetics does this through chemical transformations. There is also a change in background temperature associated with each: molecular diffusion by moving hot or cold material from one region to another, and chemical kinetics by releasing or absorbing energy. These interactions are particularly important in flames.

2–4. Complications of Multiphase Flow

Multiphase flow describes a broad category of problems, including the behavior of grain or coal dust suspended in air, droplets and sprays injected into a gas, propellant burning in engines, charring, soot, smoke formation, slurries, bubbles in liquids, rain, sedimentation, dust in the earth's atmosphere or interstellar medium, sand in water, stars and gases in galaxies, and even galaxies and dark matter in the early universe. Each of these systems has distinguishing characteristics that keep any particular multiphase model from being generally applicable. The result is that many disjoint modeling communities use their own specific formulations and approximations.

One approach is to use a basic set of Navier-Stokes equations, and then graft extensions onto these that represent other multiphase physical interactions and system properties. This is similar to what has been done to generate the multispecies reactive-flow equations given at the beginning of this chapter. In the case of multiphase flows, this approach is sometimes done in a rather *ad hoc* manner using physical intuition and phenomenologies. A potential problem with this approach is missing important interaction terms. In general, the choice of what to treat from first principles and what to represent through a simplifying phenomenology is not always obvious. Excellent discussion of multiphase flow can be found in the various books by Soo (1967, 1984, 1990), Kuo (1986), and Crowe, Sommerfeld, and Tsuji (1998), and in review articles, for example, by Crowe, Troutt, and Chung (1996) and Faeth (1996).

2–4.1. Interactions among Phases

The terms in the multiphase reactive-flow equations resemble those described above for multispecies reactive gas flow. The equations now contain new coupling terms between the phases and new phenomenologies added specifically to describe multiphase processes. There are four major types of differences that occur in the properties of coexisting phases:

1. *Chemical Differences.* Because chemical reactions among species in different phases locally change the numbers of molecules in each phase, it is often necessary to treat the phases and species individually. For some multiphase flows, a different equation represents the same species in each phase.
2. *Thermal Differences.* The different phases can have different temperatures even though they may occupy the same macroscopic fluid element. This occurs because velocity equilibration can be much faster than temperature equilibration. If each

phase or component is described by a different temperature, the number of equations to be solved increases. With temperature differences among the phases, coupling terms are needed to describe energy transfer. The temperature coupling equations often involve terms of the form $(T_i - T_j)$, where the indices i and j refer to different phases or components of the composite medium.

3. *Dynamic Differences.* Sometimes it is necessary to describe each phase by its own average velocity field. This introduces new coupling terms proportional to $(\mathbf{v}_i - \mathbf{v}_j)$. New phenomena may arise from differential accelerations on drops or particles in a fluid. The *effective mass* of the particle appears larger than the actual mass because the background fluid must be displaced when the particle or droplet accelerates through it.
4. *Spatial Separations.* Sometimes we need to consider the spatial extent of the different phases. With different phases in the same region, we cannot always assume that they completely interpenetrate and occupy the same volume. This was not a significant problem in multicomponent gases. When the spatial separation between phases is not completely resolved in the model, it might be necessary to consider what fraction of the volume is occupied by each material. A much more fundamental but more difficult approach is to resolve the spatial separation and consider the material interfaces.

2–4.2. An Example of Multiphase Flow Equations

The four types of differences between coexisting phases listed in Section 2–4.1 have given rise to a myriad of problem-dependent modeling approaches. These range from the simplest case of a single-fluid model, in which the different phases are considered a single fluid whose properties are composites of those of the various phases present, to multifluid models, for which the different phases travel at different velocities that may deviate substantially from the mean flow velocity. For example, flows containing very small droplets or particles can often be treated as a single fluid, but rain requires a multiphase model because there is significant relative motion between the air and the falling water drops. Here we describe one example of a multiphase flow model, based on the Navier-Stokes formulation, and use this as a starting point to discuss some of the complexities.

For this discussion, we omit the effects of chemical reactions and radiation transport. Let the different phases be labeled by the subscripts i, j, ... Then

$$\frac{\partial \wp_i \rho_i}{\partial t} + \nabla \cdot \wp_i \rho_i \mathbf{v}_i = \Gamma_i \tag{2–4.1}$$

$$\frac{\partial (\wp_i \rho_i \mathbf{v}_i)}{\partial t} + \nabla \cdot (\wp_i \rho_i \mathbf{v}_i \mathbf{v}_i) + \nabla \cdot (\wp_i \mathbf{P}_i) = \mathcal{M}_i - \mathcal{F}_i \tag{2–4.2}$$

$$\frac{\partial \wp_i E_i}{\partial t} + \nabla \cdot (\wp_i E_i \mathbf{v}_i) + \nabla \cdot (\wp_i \mathbf{v}_i \cdot \mathbf{P}_i) + P_i \frac{\partial \wp_i}{\partial t} + \nabla \cdot \mathbf{q}_i = \mathcal{L}_i, \tag{2–4.3}$$

where the $\{\Gamma_i\}$, $\{\mathcal{M}_i\}$, $\{\mathcal{F}_i\}$, and $\{\mathcal{L}_i\}$ are coupling terms among the phases. The $\{\Gamma_i\}$ are

mass transfer terms representing the rate of production of phase i. The effect of mass transfer on the momentum equation is given by $\mathcal{M}_i$, which is proportional to Γ_i. The effects of external forces comes into the momentum equation through the term $\{\mathcal{F}_i\}$, which might correspond to an interphase frictional drag term proportional to $\mathbf{v}_i - \mathbf{v}_j$. The energy coupling, $\{\mathcal{L}_i\}$, is composed of several terms, one proportional to Γ_i, one related to $\mathcal{F}_i$, and one proportional to $(T_i - T_j)$,

Besides the new coupling terms, there is a new variable here, φ_i. The physical meaning of this quantity and any equations describing its behavior depend on the specific problem. It is usually used to represent the volume fraction occupied by a particular phase. The mixture quantities are then, for example,

$$\rho_m = \sum_i \varphi_i \rho_i \tag{2–4.4}$$

$$\rho_m \mathbf{v}_m = \sum_i \varphi_i \rho_i \mathbf{v}_i, \tag{2–4.5}$$

The scalar pressure P is often assumed to be equal in the various phases, although the viscosities μ_i that would appear in a pressure tensor $\mathbf{P}_i$ may not be. If surface tension is present, the pressures of the different phases are not equal, as indicated by the P_i in equations (2–4.1) through (2–4.3). Pressure equilibrium and conservation constraints help to define the values of various input coefficients, but most of the basic physics of the problem lies in the choice of coupling parameters and which processes and interactions are assumed negligible.

Extending this model to reacting flows would require adding sets of equations of the generic form for equation (2–1.2) for the number densities $\{n_k\}$, where now the chemical species k may be in different phases. The presence of chemical reactions would add further complexity to computation of $\{\mathbf{q}_i\}$ and $\mathbf{P}_i$.

2–4.3. The Complexity of Multiphase Models

To formulate a multiphase flow problem, it is necessary to consider the type and extent of interactions among the phases. It is helpful to know ahead of time whether dynamic equilibrium and temperature equilibrium are valid assumptions. To help determine this, it is useful to evaluate the level of *collisionality* between the phases. When the flow is *fully collisional*, all the phases move together and a single macroscopic velocity field is adequate to describe the motions. This simplification, called a *single-fluid model*, is used frequently, even when it is not strictly valid, because treating only one flow field greatly reduces the computational cost.

When phases are in thermal and dynamic equilibrium, a single temperature and a single-fluid velocity are adequate. The separate phases behave rather like separate chemical species. Extensions of single-fluid models are common and useful. For example, it might be possible to consider small differences in velocities of the individual phases about the mean flow velocity. This approach allows a more complete treatment of phase interpenetration. It might be possible to consider temperature differences between the phases. Multitemperature models cover a range of problems, and sometimes it is necessary to

consider separate electron, neutral, and ion temperatures, as in the case of models of the earth's upper atmosphere, the sun, and other high-temperature, nonequilibrium systems. The multitemperature generalization leads to coupling terms between the two energy equations of the form $(T_i - T_j)$.

When there are not enough collisions to equilibrate the flow, it might be possible to describe each phase by its own velocity field. This is called the *multifluid approximation*. Self-consistent calculations of multifluid models must be used in cases when the relative velocities of the phases are comparable to the mean flow velocity. One example of such flows is heavy or rough particulates in a background gas, such as soot particles in air when the soot may be represented as a single species. Another example is sand in a sandstorm. Using two momentum equations in this case might allow us to calculate the way gravity pulls the sand out of the high-speed winds.

When the phases are collisionless, or nearly collisionless, the use of local dynamic or thermodynamic equilibrium for each of the collisionless interpenetrating components is hard to justify. In fact, using a continuum momentum model itself is open to question. Particle models, as opposed to fluid models, are often necessary to represent the full complexity of a truly collisionless medium.

2–5. Finding Input Data

Table 2.5 is a summary of equations (2–1.1) through (2–1.19) written in terms of the physical processes and the required input information. Obtaining good values for the input data to the equations is as important as using good numerical algorithms to solve them. There may also be as much difficulty in determining these values as in building the simulation model. There are often no convenient tables or standard references, and extensive literature searches may provide ambiguous results. Publication of acceptable data may be in obscure sources, or its interpretation may require extensive computation.

The result is that the input data are usually the least accurate parts of the computation. Both convection and diffusion processes can usually be computed to a few percent accuracy. The transport coefficients and reaction rate coefficients are seldom known to 10 percent accuracy. There are situations, however, when a process is extremely important

Table 2.5. Processes in Conservation Equations

Process	Inputs	Chapter
Chemical kinetics	Species and reactions, enthalpies, heats of formation	5
Molecular diffusion	Molecular diffusion coefficients	7
Thermal diffusion	Thermal diffusion coefficients	7
Viscosity	Viscosity coefficients	7
Thermal conduction	Thermal conduction coefficients	7
Convective transport	Boundary and initial conditions	10
Radiation transport	Absorption and emission coefficients	13

to a large enough (or wealthy enough) community, and so efforts are made to determine the input to sufficient accuracy. This is currently the situation, for example, with selected hydrocarbon reactions needed to produce soot, and has been true for certain groups of atmospheric chemistry, thermonuclear, and atomic reactions. It is often useful to benchmark a code with a well-known set of reactions before proceeding to extensive and ambiguous reaction mechanisms.

A problem that often arises in chemical reactions is that there are fundamental inconsistencies in a measured reaction rate. For example, there may be experimental measurements of both the forward and the reverse rate constants, k_f and k_r. Nonetheless, when either is combined with the equilibrium coefficient for that reaction, the other is not produced. This appears to represent a violation of equilibrium thermodynamics. The explanation is usually that k_f and k_r have been measured at rather different temperatures or pressures, and so there are inconsistencies when they are extrapolated outside the regime of validity of the experiments.

REFERENCES

Brodkey, R.S. 1967. *The phenomena of fluid motions*. Reading, Mass.: Addison-Wesley.

Chabrier, G., and E. Schatzman. 1994. *The equation of state in astrophysics*. Cambridge: Cambridge University Press.

Chapman, S., and T.G. Cowling. 1970. *The mathematical theory of non-uniform gases*. 3rd ed. Cambridge: Cambridge University Press.

Crowe, C.T., T.R. Troutt, and J.N. Chung. 1996. Numerical models for two-phase turbulent flows. *Ann. Rev. Fluid Mech.* 28:11–43.

Crowe, C., M. Sommerfeld, and Y. Tsuji. 1998. *Multiphase flows with droplets and particles*, Baton Rouge, LA: CRC Press.

Eliezer, S., A. Ghatak, and H. Hora. 1986. *An introduction to equations of state: Theory and applications*. Cambridge: Cambridge University Press.

Faeth, G.M. 1996. *Spray combustion phenomena*. In *twenty-Sixth Symposium (International) on Combustion*. pp. 1593–1612. Pittsburgh: The Combustion Institute.

Gathers, G.R. 1994. *Selected topics in shock wave physics and equation of state modeling*. Singapore: World Scientific.

Hirschfelder, J.O., C.F. Curtiss, and R.B. Bird. 1964. *Molecular theory of gases and liquids*. New York: Wiley.

Incropera, F.P., and D.P. DeWitt. 1990. *Fundamentals of heat and mass transfer*. 3rd ed. New York: Wiley.

Landau, L.D., and E.M. Lifshitz. 1959. *Fluid mechanics*. New York: Pergamon Press.

Kippenhahn, R., and A. Weigert. 1990. *Stellar structure and evolution*. Berlin: Springer-Verlag.

Kuo, K.K., 1986, *Principles of combustion*. New York: Wiley.

Mason, E.A., and T.H. Spurling. 1969. *The virial equation of state*. New York: Pergamon Press.

Mihalas, D., and B.W. Mihalas. 1984. *Foundations of radiation hydrodynamics*. New York: Oxford University Press.

Modest, M.M. 1993. *Radiative heat transfer*. New York: McGraw-Hill.

Reid, R.C., J.M. Prausnitz, and T.K. Sherwood. 1977. *The properties of gases and liquids*. New York: McGraw-Hill.

Rybicki, G.B., and A.P. Lightman. 1979. *Radiative processes in astrophysics*. New York: Wiley.

Siegel, R., and J.R. Howell. 1993. *Thermal radiation heat transfer*. Washington, D.C.: Hemisphere.

Soo, S.L. 1967. *Fluid dynamics of multiphase systems*. Waltham, Mass.: Blaisdell.

———. 1984, Development of dynamics of multiphase flow. *Int. J. Sci. Eng.* 1:13–29.

———. 1990. *Multiphase fluid dynamics*. Beijing: Science Press, distributed through Gower Technical, Brookfield. Mass.

Strehlow, R.A. 1984. *Combustion fundamentals*. New York: McGraw-Hill.

Williams, F.A. 1985. *Combustion theory*. Menlo Park, Calif.: Benjamin Cummings.

Zeldovich, Ya. B., and Yu. P. Raizer. 1966–1967. *Physics of shock waves and high-temperature hydrodynamic phenomena*. New York: Academic.

3

Models and Simulation

The most effective use of any simulation model is determined as much by the research and computing environments in which it is used as by the particular models and algorithms that happen to be included. Therefore, before considering specific algorithms and models, it is important to consider the processes of developing and using simulation models and to relate them to the computing environment where the work will be done. The information in this chapter, which focuses on constructing and testing simulation models, is just as important to those using a code written by someone else as to those writing their own reactive-flow codes. Knowledge of the considerations and trade-offs involved in building a model can help the user detect and understand the source of apparent errors and the costs or feasibility of fixing them. Without the experience gained from having actually built the model, a user can be too easily lulled into a false sense of accuracy and reliability, and thus completely baffled by the results of a computation.

There are three main stages in developing and using a numerical simulation model. Each stage has three aspects:

1. Specifying the mathematical and computational model
   - Posing the scientific reactive-flow problem
   - Selecting the computer system
   - Choosing a computational representation
2. Constructing the numerical simulation model
   - Choosing suitable algorithms for the problem and the representation
   - Optimizing these model algorithms for the problem and computers
   - Installing and testing the model on the computer system
3. Using the numerical simulation model
   - Performing detailed simulations in a careful, organized way
   - Analyzing, understanding, and documenting the results
   - Optimizing the overall detailed simulation activity

This chapter gives a general overview of the first two stages and a number of suggestions for the third. Much of the material covered here is only "learned" in a practical sense by experimenting computationally and coping with the consequences.

3–1. Developing a Numerical Simulation Model

3–1.1. Posing Scientific Problems for Modeling

The first step in developing a numerical simulation model is to pose the scientific questions in terms that can be answered effectively by a simulation. For example, we may need to evaluate how dangerous it is to use a match near the vent of an automobile fuel tank while refueling. In more scientific terms, we may ask: How likely is it for the combustible, but rapidly diluted gases escaping from the vent to ignite or detonate? Can any combustion that may occur propagate past the nozzle into the automobile's fuel tank? To help answer these questions, we must ask related but more specific questions about the minimum ignition energies, the flammability and detonation limits, and the flame speeds of gas mixtures which are composed of gasoline fumes diluted with various amounts of air, and so on. Simulations can now be expected to provide answers to these specific questions, which, in turn, provide information that helps quantify the safety of refueling a car.

The underlying question to ask is: *What do you intend to learn about the real system from a simulation?* This question has two aspects. First, you have to accept that what you *intend to learn* can change dynamically as the study progresses. Because the problem-solving process is begun with limited knowledge, expect to ask some wrong questions initially, or perhaps to ask the right questions but from the wrong perspective.

Second, the constraints imposed by our computational tools have a controlling effect on the problems we can study. The best way to get answers to problems quickly is to identify and work well within the constraints of the existing simulation capability. Otherwise, it is necessary to have the time, experience, and resources to undertake an extensive development project. For example, the fact that the refueling nozzle is usually introduced to the tank opening at an angle will probably be unimportant for a first cut at the problem. Its inclusion requires a full three-dimensional treatment, so it might be better to start with a simpler axisymmetric, two-dimensional geometry.

Often finding answers involves simulating an interaction among several physical processes as a parameter in the model is varied. You might ask how the minimum ignition energy of a mixture of 10 percent air and 90 percent gasoline vapors changes as small amounts of realistic additives are introduced. To answer this question, the computer program that implements the model must represent the interacting effects (chemical reactions and fluid dynamics) at the right level of accuracy, in all the regimes of pressure and temperature where they are important for this problem, and over a range of model parameters (for example, the percentage of octane in the mixture). By focusing on detailed physical interactions suspected to be of primary importance, less relevant aspects of the real system can be treated in an idealized way or perhaps omitted.

Listing clearly and precisely what information is needed to solve a very specific "standard" problem makes planning the simulations much easier. This standard problem should be chosen to focus attention in the middle of the parameter regimes of interest and should be based on the physically relevant sizes and conditions. It becomes a representative and relevant standard case. Having such a standard case for the research project can be used to organize the development of a simulation model and help determine what model,

algorithmic, and diagnostic modifications may be needed. The standard case should be as idealized as possible and yet still be related closely to a specific problem. Too many computational projects are started with the goal of developing a general model capable of accurate prediction over a wide parameter range but without a particular application in mind. When the focus of a simulation project is too diffuse, little useful or practical information is gained compared to the effort expended. For example, a standard case for the automobile refueling problem might be: How close to the refueling nozzle can a match be safely lit when there is no wind and a steady stream of 100 percent octane vapor is being displaced by a fluid level rising at a five-gallon per minute fill rate?

The detailed simulation of the standard case should produce answers in physical units corresponding to the most relevant experimental or analytical results available. By the time the standard-case simulation is completed satisfactorily, the numerical model includes the relevant physical processes and the numerics in the problem have been checked. The results of the standard case can be calibrated and validated against existing experimental data and compared to results of subsequent simulations in which different parameters of the system are varied. For example, the sensitivity of the ignition energy to the background air temperature could be checked by changing the air temperature and repeating the simulations. The sensitivity to the grade of the fuel (gasoline) could be checked by evaluating how the safe distance for igniting the match changes as the percentage of higher hydrocarbons is increased. When new or modified algorithms are incorporated in the simulation, the standard case provides a familiar basis for evaluating potential improvements or generalizations. For example, a new integration algorithm can be used, and the results compared with the standard case.

The specific problem and the standard case do not have to model the real system exactly to be useful. It is usually best to solve a slightly different problem well rather than do a poor job on the exact problem. By leaving out many of the components of a complex system and focusing on only a few interactions, we can often formulate a computational problem whose predictions help our understanding of the real system. The simulation might not have to reproduce the complicated geometry of the station in which the actual refueling occurs. Then we might be able to calculate the ignition energy in planar and spherical geometries to bound the answers. If a main effect is controlled by the geometry, however, the calculation should be multidimensional.

3–1.2. Selecting the Computer System to Use

Computers are becoming faster, smaller, and cheaper. The large computers generally available for scientific and engineering work, generally new architectures with multiple processors, are approaching 10^{12} useful arithmetic floating-point operations per second (flops). More conventional supercomputers with a few processors and modest-sized parallel systems can deliver about 10^{10} operations per second. Moderately priced, desktop workstations are becoming available with speeds in excess of a gigaflop (10^9 floating-point operations per second). Even inexpensive desktop systems now deliver hundreds of megaflops with a wide range of relatively inexpensive, robust, user-friendly software. This means that a user wanting to do a detailed reactive-flow simulation has a

wide range of options spanning about four orders of magnitude. Useful perspectives on the growth of computers and issues in selecting a computer are given by Landau and Fink (1993).

What can be accomplished with the computing resources under the user's direct control is quite impressive. The cost of three-dimensional, time-dependent simulations generally scales as the fourth power of $L/\Delta x$ (where L is the size of the computational domain being modeled, and Δx is the size of the cells into which the domain is divided for the simulation). Therefore, the largest computer in existence can solve a problem that is less than an order of magnitude better resolved than a good personal computer. There are millions of powerful small computers in use today while there are only about a dozen delivering 100 gigaflops or more. Thus over 95 percent of all computing is done on small computers.

The small computers are cheaper per computation because so many of them are manufactured. Further, their software is generally more reliable and easier to use than the software for the most recent, state-of-the-art multiprocessor. This means that using the largest available computer is not necessarily the best approach. At a regional or national supercomputer center, a user shares a modest resource with a large number of people who feel that bigger is better. Once the computing environment has hundreds or thousands of other users, the computational throughput is effectively out of the user's control because of long queues, competition for resources, and priority scheduling with conflicting agendas.

The very biggest computers are also usually the very newest. Today, these computers are generally built around the notion of parallel processing – solving a single problem with many processors working simultaneously. The term *scalable processing* has come to mean programs written so the user can scale up the number of processors to fit bigger problems without reducing the efficiency with which each processor tackles its part of the overall job.

The networking and input/output communications for the new, scalable computers are generally not reliable for the first year or two. Their compilers are relatively new and full of bugs. They are more difficult to use than less impressive computers closer to home, where users have already encountered and solved the problems. A wider range of useful, auxiliary software will also be available for the smaller systems. Further, a massively parallel scalable system is intrinsically more difficult to use than the smaller systems. System and hardware specialists are constantly taking these computers away from the users to work on them, while computer scientists and manufacturers, who have no vested interest in getting a reliable, accurate answer in a limited amount of time, will pressure you to reprogram your software in new, untested languages.

A good rule of thumb is to use readily available computers that are about a half generation (two to three years) behind the current state of the art. These older, more tested computers work reliably and well, and the computer scientists and engineers are no longer interested in them. Whenever possible, you should always have several computing options, for example, the workstation at your desk and a larger system shared with ten to a hundred other users. Staying familiar with two or more systems will allow you to better select the right computer for each portion of the project. It will also encourage programming and computing practices that ease the transition of "porting" the simulation model to third and fourth computer systems as that inevitably becomes necessary.

3–1.3. Choosing the Computational Representation

Constructing Mathematical and Computational Submodels

In Chapter 1, we described the importance of chunking in theoretical models of physical systems. The statistical nature of many of the usual physical models gives meaning to relatively large-scale averages such as density and temperature. Chunked or averaged models are useful because it is not usually practical to solve complicated problems at the most microscopic level of detail. An averaged representation is often adequate.

The most natural way of structuring a reactive-flow computer program is by combining software modules, each of which represents a different physical or chemical process. Each of these modules has, in fact, a representation that arises from chunking more detailed representations of that particular process. (Examples of such modules could be fluid dynamics, chemical reactions, diffusion, etc.) The selected set of chunked models is then coupled to create the complete simulation model. In this way it is easier to replace a submodel with a more detailed submodel, if it is ever needed. The importance and methods for maintaining this type of modular approach is a theme of this book.

The Representation Affects the Answers

The *representation* of a physical system includes the mathematical form of the equations, the discretization used, and any auxiliary conditions needed to describe the problem. Sometimes a mathematical or computational representation is adopted and the assumptions that justify its use are ignored or forgotten. In such cases, the simulation results can be prejudiced by the representation in surprising ways. Because the representation establishes the framework of the model, it affects the answers both directly and indirectly. For example, a reactive-flow model might predict a temperature field with an apparently reasonable value for each computational cell. Even so, when the representation excludes time variations, the results will be incorrect when there are very rapid transients. Determining a suitable computational representation for the complete set of equations and for the various physical submodels is the next important step after choosing the problem to solve.

The Navier-Stokes equations are a reasonable starting point for calculating the ignition energy of the gasoline-air mixtures in the problem discussed above. Using these equations assumes that changes in physically averaged variables over a mean free path are small, so the mass, momentum, and energy fluxes through a volume element can be related by first derivatives to the density, velocity, and temperature. Once the Navier-Stokes representation is chosen, however, meaningful descriptions of processes with large gradients over a mean free path are excluded. The chunked Navier-Stokes model cannot predict the detailed structure of a shock, whose width is determined by the viscosity term in the Navier-Stokes equations or by unphysical smoothing effects in the numerical algorithm chosen.

As another example of how the representation can prejudice the answer, consider a statistical turbulence model describing the time-averaged flow properties of a turbulent jet. Such models contain coefficients that are fit to experimental data or to some specific theory. They will probably predict the macroscopic fluid quantities well for parameters close to those used to develop the model. Nonetheless, such statistical turbulence models cannot predict deterministic local values or phenomena such as the shedding and breakup of large

eddies. The model also cannot tell us anything new and fundamental about turbulence. It is dangerous to use a model outside its calibrated parameter regime or to use it coupled to models of other physical effects. Even though the limitations may be clearly stated, there is a tendency to expect a model to be better than it is, particularly when it appears to be generating reasonable numbers.

It is possible to include essentially all of the macroscopic consequences of the actual atomic and molecular phenomena in a fluid model. Sometimes, however, the presence of terms describing these microscopic processes in the macroscopic representation can lull us into thinking that we have a detailed enough representation to make a prediction. For example, in calculating ignition energies for a venting vapor mixture, molecular diffusion is one of the macroscopic continuum effects that must be included. On the microscopic level, a random walk transports the lighter, hotter species ahead of the flame front and prepares the mixture for ignition. If we use the standard second-derivative form to represent molecular diffusion, we are assuming that a continuum representation is valid. If the random walk were not well represented by a diffusion equation, the ignition process would not be properly represented.

An important part of the fluid description of localized ignition is the value of the effective molecular diffusion coefficients. Because it is hard to measure diffusion coefficients over the wide pressure and temperature ranges encountered in combustion systems, estimated and extrapolated values are often used. For example, if a molecule is polar, the molecular interaction from which the diffusion coefficient is derived should reflect the property of polarity. Otherwise, the diffusion coefficient will probably be wrong. Then the diffusion process might not be treated properly even though microscopic diffusion appears to be well represented.

Finally, a lumped-parameter chemistry representation can often be derived from a detailed chemical kinetics mechanism involving many elementary reactions among many chemical species. This is a calibrated phenomenology that can calculate, but not really predict, chemical induction times, energy release rates, and equilibrium conditions, given initial temperatures and pressures. A lumped-parameter chemistry model interpolates the known results in regions where it has been calibrated. In regions where it has not been calibrated, the accuracy of the answers is unknown, even though the values given by the model may appear to be reasonable.

These examples illustrate how the physical and numerical representation affects the validity of the answers. When developing a model and deciding whether it can be used to solve a particular problem, it is important to keep in mind that *a computer simulation cannot correctly describe an effect if the controlling physical processes are not adequately represented.* Remember which physical processes and interactions have been approximated or removed from the equations, so that a crucial process is not left out. This is obvious advice, but the results of forgetting it are insidious.

3–2. Limiting Factors and Constraints

There are a number of additional limitations and constraints to cope with beyond those introduced by the choice of the problem representation. To integrate the model numerically, we must discretize the equations in both time and space. This gives a consistent,

computable representation. Space is generally divided into finite-sized parcels called *computational cells*, or sometimes *zones*. Time is divided into finite intervals called *timesteps*. The approximate numerical solution in each of these computational cells is advanced one timestep at a time. It is important to make sure that the computation actually implemented on the computer has the correct continuum limits. The solution obtained numerically should approach the solution of the original continuum equations as the timesteps and the cell size are decreased.

In practice we can seldom approach the continuum limit as closely as we would like. The accuracy and regime of validity of computational simulations are limited by the number of cells and timesteps that can be used and by the accuracy and stability of the numerical methods. These resolution-limiting, accuracy-limiting factors are imposed primarily by computer hardware constraints such as computer memory, computation speeds, precision data formats, and data storage facilities.

Another type of computational limitation is the high cost associated with solving complex problems. The limitations arising from complexity and from accuracy are intertwined and often hard to distinguish. The physical and geometric complexity of a problem can make the cost of constructing and carrying out simulations prohibitive. These complexity costs are based on personnel and computer costs and the time involved in developing and maintaining reliable software, rather than the performance of the computer hardware. Complexity leads to bigger models with more special cases to program and test. The likelihood of undetected errors, or "bugs," increases. Therefore, these costs often appear as software constraints. Complexity means more initial data must be found and cross-checked. It means also less efficient use of the computational hardware and slower-running programs.

We can identify three types of accuracy limitations brought on by hardware constraints: temporal, spatial, and computational. We can also identify causes of costs due to the complexity of the problem: physical, geometric, and computational. Below we discuss these limitations, and explain how phenomenologies are used in order to circumvent some of them.

3–2.1. Limitations in Accuracy

Temporal Accuracy Limitations

Limitations in temporal accuracy arise from trying to represent different, widely separated time scales simultaneously. Physically important time scales can range over ten or more orders of magnitude in flame and detonation problems. As long as the relevant time scales can be resolved by the numerical method, obtaining the desired simulation accuracy is not necessarily a limitation. Computing for more timesteps takes longer, but does not cause the problem to exceed the computer memory. Cost, however, can quickly become prohibitive.

When we wish to model processes with characteristic times shorter than the timestep we can afford, we sometimes refer to the equations that describe these phenomena as "stiff." This is a common use of a word that in some cases has more precise mathematical definitions. This is shown in Chapter 5 in the discussion of the various types of stiffness that can occur in ordinary differential equations. More generally, sound waves will be stiff when the timesteps of a calculation are well suited for modeling a laminar flame, because resolving the sound waves computationally usually requires a much shorter timestep than

modeling the propagation of the flame. The equations describing many aspects of complex combustion chemistry are stiff with respect to convection, diffusion, or even the characteristic timestep based on the speed of sound in the material. Stiffness is sometimes a matter of practical timestep limitations, but can also be an inherent property of certain types of equations. Two distinct modeling approaches, the global-implicit approach and the timestep-split approach, have been developed to treat temporally stiff phenomena. These two approaches are discussed in more detail in Chapter 11.

Spatial Accuracy Limitations

There are also limitations that arise from the need to resolve a wide range of space scales. In continuum representations with spatial gradients, greater accuracy comes with finer resolution in both space and time. If we want to calculate the behavior of a small region in the flow with even moderate accuracy, there must be a minimum of three or four computational cells spanning the region. If steep gradients are involved, more cells are usually necessary. Structures that are not resolved well are "numerically diffused," as will be explained in Chapter 4. Because the computational cells have a finite size, it is uncertain exactly where in the cell a smaller feature of the flow actually lies. This uncertainty decreases as the spatial resolution increases.

To resolve steep gradients in a one-dimensional propagating flame front, for example, computational cells of 10^{-3} to 10^{-2} cm might be required. On the other hand, cells of 1 to 10 cm size might be adequate to model convective transport. A relative factor of 10^3 means that this spatial resolution problem probably cannot be eliminated just by making the data arrays bigger. The computer itself might not be adequate for such a calculation, especially if the problem is three-dimensional and requires 10^{-3} cm resolution everywhere.

Limitations on spatial resolution can be more severe than those of temporal resolution because the timestep, needed to maintain a stable, accurate solution, is tied to the size of the computational cells. Doubling the spatial resolution in a one-dimensional calculation usually requires a fourfold increase in computation to integrate a given length of time. One factor of two comes from the additional cells, and another factor of two is caused by the halved timestep required to integrate the equations using these smaller cells. Doubling the resolution in two dimensions requires a factor of eight times as much work to advance the calculation the same physical time. In three dimensions, a factor of sixteen times as much work is required. Therefore, algorithms that give accurate answers on coarse grids are extremely valuable.

When the space scales to be resolved differ greatly, adaptive gridding methods may be used to put spatial resolution only where it is needed (see Chapter 6). A still more difficult problem is encountered in turbulent flows, in which a large range of spatial scales may be important throughout the flow field. In general, we seek numerical methods that maximize accuracy with a minimum number of grid points. Even with the best algorithms, approximations and phenomenologies are required to represent phenomena that occur at subgrid scales.

Computational Accuracy Limitations

Accuracy limitations arise from trying to represent the continuous functions of the fluid variables in time and space. We are forced to assign finite-precision, real numbers to

describe the properties of these fluid elements. Even though the representation for each number may be valid to seven, or fourteen digits, round-off errors will still accumulate. For example, a 32-bit floating-point number is accurate to one part in a few million. Calculations with many thousands of timesteps often show accumulated round-off errors in the third decimal place with 32-bit numbers. Other types of hardware problems, such as "underflows" and "overflows," can arise in arithmetic calculations where the product of two numbers is too small or too large to be represented in the computer.

The algorithms used to implement the mathematical equations are also limited in accuracy. Evaluating the accuracy and efficiency of these algorithms is a major theme of this book. The *order* of an algorithm refers to the highest power of the expansion parameter that is retained in the solution method (see Chapter 4). The corresponding error is often called the *truncation* error. When the truncation error grows to the largest number a computer can represent, the algorithm is said to be *unstable*. When these truncation and numerical-instability properties of an algorithm interact with the precision limitations of the hardware, the accuracy is often reduced further. For example, when an algorithm is said to be *second-order accurate*, it usually means that increasing the spatial and temporal resolution by a factor of two decreases the error by a factor of four. The effective order of accuracy can sometimes be reduced due to finite precision.

Interactions of the stability properties of algorithms with the precision limits of the computer are not easily predictable. Although it is customary to view numerical stability as an algorithmic issue, numerical stability also interacts with limitations in numerical precision. Round-off errors can adversely affect stability, especially if, in the name of economy, the choice of the timesteps and cell size puts the algorithm close to the stability limit. In practice, algorithms should be chosen that are stable regardless of whether the computer hardware precision is 32-bit or 64-bit.

As computers become faster and more precise, slowly growing numerical errors, that may not have appeared previously in shorter or coarser calculations, appear and often display bizarre properties. The relatively weak effects of solution errors, algorithm instabilities, precision errors, and interactions among these, were either damped previously in coarser calculations or did not appear because the calculations were not run long enough. An example of this is round-off errors in very long calculations using 32-bit floating-point numerical representations. Single-bit errors are of little or no consequence after 1,000 timesteps, but may dominate the solution after 100,000 timesteps. The use of supercomputers allowing a factor of 100 between the largest and smallest resolved temporal scales makes these types of effects an even greater issue.

3–2.2. The Costs of Complexity

The Costs of Physical Complexity

Physical complexity costs computer time and extensive programming effort. For example, chemically reacting systems may have many interacting species. To describe the evolution of such systems correctly, the sets of coupled equations that describe them must be solved simultaneously. Complicated sets of stiff, ordinary differential equations describing reactions and large matrices describing the multispecies diffusion process are time consuming to solve on the computer. To simulate reacting flows with this level of detail

requires orders of magnitude greater calculation time than needed for idealized, empirical, or phenomenological models.

There is another cost of physical complexity: the increase in the time required to gather input data and to write and debug the more complicated numerical algorithms. The software development time can grow as the square of the number of important physical processes, since interactions among submodels can generate problems even though the separate modules may appear to be working correctly. This aspect of complexity is a software constraint.

Often the accuracy in the input data is not sufficient for the level of accuracy desired from the simulation. If much of the input data have large errors, the accuracy and level of detail asked of the simulation may be unreasonable. A much less expensive calculation with lower resolution might do as well. On the other hand, if only some of the data are poorly known, the simulations can be used in conjunction with relatively more accurate empirical data to help determine the uncertain input parameters.

The Costs of Geometric Complexity

Realistic systems often have complicated geometries. The spatial, temporal, and computational problems associated with modeling complicated geometry can, in principle, be solved by larger, faster computers. There are again software and personnel time costs which arise from the need to represent, program, and diagnose the geometric complexity. These costs are not decreased by improvements in the computer system. Visualizing solutions in complex geometry is also more difficult, time consuming, and costly than the corresponding diagnostics for simpler flows.

It takes more time to program a system with complex geometry than one with simple geometry, and there is more computer code to go wrong. Even if the program runs properly the first time, it will take longer to run and more time to get to the point where there are worthwhile results. It will require more computer storage, and the output will be harder to interpret. When enormous software structures are built, they tumble down very easily. Software complexity is an aspect of scientific computing now receiving considerable attention by computer scientists and software engineers.

Geometric complexity often requires increased dimensionality and more complex boundary conditions. For example, experimental probes in a reactive-flow act as obstacles, and their effects must often be modeled. In the absence of a probe, port, or exhaust valve, a one- or two-dimensional model might be adequate. Realistic geometric complications force a reactive-flow problem to be represented in at least two and probably three dimensions. Thus a major part of experimental and computational ingenuity is often directed toward finding a meaningful problem of lower dimensionality.

One-dimensional models are often used, even though they usually give only a semiquantitative assessment of the physical process. Practical combustion systems, including real flames and detonations, generally involve multidimensional effects such as transverse waves, boundary-layer growth, formation of vortices, separating flows, turbulence and internal sources and sinks. These systems require two- and three-dimensional fluid dynamics. Even with parallel-processing supercomputers, what can be achieved with a three-dimensional detailed model is limited by computer time and storage requirements. Here again, cost usually limits accuracy.

In situations with complicated boundary conditions and higher dimensionality, a favorite algorithm may no longer work properly. To include the effects of holes, bumps, and knobs on a surface, algorithms must be flexible enough to treat unusual boundary conditions or generalize to multidimensions. Abandoning good algorithms for less accurate but more general ones is a major cost of geometric complexity. Making these trade-offs is a fundamental consideration in building simulation models, and thus it is a major theme of this book.

The Costs of Computational Complexity

Computational complexity is more than the result of implementing physically and geometrically complex problems in a simulation model. The need for model flexibility in the face of a wide range of physical input conditions and for portability between different computer systems makes programming and testing the program more difficult. Allowing for input/output procedures suitable for different computers may make the process of starting or restarting a simulation much more costly. Not only the computer running time but also personnel costs for developing, maintaining, and running the simulation model become large. In addition, numerical problems with the computation occur more often because there are more places where the errors might be and more reasons why they might occur. As a simulation model becomes larger and more comprehensive, it becomes more difficult to verify and modify. The software then becomes a serious constraint. A careful, modular approach to model development with many standard test problems is the only reasonable way to proceed.

3–2.3. Using Phenomenological Models

The list of costs and limitations encountered in developing and applying simulations is formidable. To deal with them, we can improve our numerical algorithms and other software, and we can take advantage of larger, faster, more accurate, and friendlier computers. Over the last few decades, improved algorithms have contributed to simulation capability as much as hardware development. Together, hardware and software advances have increased our capability by orders of magnitude. Early one-dimensional gas dynamics calculations were resolved with forty or fifty computational cells. Today, three-dimensional calculations, which require five or six orders of magnitude more computing, take about the same amount of computing time as the early one-dimensional calculations.

Important algorithmic advances in most of the reactive-flow disciplines are coming at wider intervals as the technology matures. Moreover, since the new parallel computers are harder to use effectively than the workhorse, single-processor supercomputers of the previous decade, the overall rate of progress has slowed further. Thus qualitative leaps in our ability to solve realistic problems are more costly and now few and far between.

A way to deal with resolution and complexity limitations and costs is to isolate the expensive processes and to replace them in the simulation with simplified phenomenological models. As discussed in Chapter 1, these phenomenologies are *ad hoc*, approximate, physically reasonable models with parameters that are taken from theoretical estimates, based on fits to experiments, or calibrated from detailed but more expensive numerical

models. As a rule of thumb for constructing phenomenological models, the more comprehensive the underlying theoretical framework for the phenomenology, the smaller the tables of precomputed results needed and the broader the validity of the model.

Phenomenologies can be dangerous, however, because there is always a tendency to overestimate how much they can actually predict. Phenomena cannot be predicted accurately when the controlling physical processes are not resolved well enough in time, in space, or by the computational model. If one of the controlling physical or chemical processes in a simulation is being treated by a phenomenological model, the entire simulation may be no more accurate than the phenomenology, even if the other effects are treated by more detailed models.

Phenomenologies are extremely valuable in detailed modeling when you know that they are not the least accurate part of the overall model or at least that they do not obscure the reactive-flow processes we are trying to study. In this case phenomenologies allow us to focus the computational power directly on the scientific or engineering questions at hand. The spatial resolution can be increased, for example, when a simplified chemical-reaction model with just a few species, or based on tables of induction and energy-release times, is used in place of a detailed, hundred-species mechanism. Generally, a difficult reactive-flow problem has three or four distinct physical processes interacting, either all at once or in different places and different times in the flow. Processes defined by the solution of partial differential equations are generally harder to represent than those that are not. Computationally, it is relatively hard to simulate any of these processes using detailed models and relatively easy using reduced-complexity, phenomenological models.

When combining models of distinct processes, keep in mind that the work will be significantly simplified by judicious use of phenomenologies, even though the task of calibrating and verifying the model will be correspondingly more difficult. An application with either a phenomenological fluid component or a phenomenological chemistry component, allows the computational power to be concentrated on the harder aspects of the problem. Today detailed modeling of both chemical and fluid processes together is possible for many problems whenever additional processes, such as radiation transport or soot formation, can be made relatively easy by phenomenologies. A rather realistic methane-air chemistry can be used in simulating a propagating flame or detonation in one or two spatial dimensions, for example. No new technology is needed, but hundreds of hours of supercomputer time are required to do this in three dimensions. A complete three-dimensional simulation may require a thousand hours of computer processor time to achieve the same accuracy as obtained in two dimensions. If a reduced-chemistry model is used, the three-dimensional problem may be soluble in ten to twenty hours. As a warning, remember that a phenomenology must always be a suspect.

Interactions between two phenomenological models are doubly suspect. We know, in principle, how to incorporate reduced chemistry phenomenologies into numerical simulations. Though not all of the input chemical rates or specific heats necessary for particular reactions or species are known, we have some confidence in the equations and how they fit into the simulation model. To represent turbulence, we often use phenomenologies with constants that are fit to experiments and other calculations. We have some confidence that the turbulence model works reasonably well in the regime in which the parameters are fit. Thus, when simulating a turbulent reacting flow, we might be computing the interaction

of two processes represented by separately tested phenomenologies. It will be no surprise that the interactions of reduced chemistry models with turbulence models has given rise to a lot of research – and arguments.

3–3. Constructing and Testing the Model

Almost every decision in constructing a numerical model involves evaluating trade-offs. Even if you have not constructed the model you are using, it is important to understand these trade-offs. For example, the computational representation should be chosen keeping in mind the various available algorithms. In turn, algorithms should be chosen keeping in mind what kind of programming techniques would be best for a particular class of computer. Scalable computers and the options for parallel processing make these choices even more interlaced and more difficult. Knowing that a good algorithm exists for treating certain processes or interactions often makes choosing the corresponding representation straightforward. The availability of a parallel computer is of little benefit if the representation or algorithms chosen involve operations or data structures that cannot be adapted to this computer architecture. Nonetheless, an algorithm that parallelizes easily might save a factor of ten in execution time, even though it requires twice as many statements and takes twice as long to program as a competing scalar or vector algorithm.

3–3.1. Trade-offs in Selecting and Optimizing Models

As computing has become more complex and the number of options has increased, the competing aspects of making the trade-offs necessary to construct a numerical simulation model have increased enormously. No one can expect to do a perfect job navigating this terrain, particularly in view of the natural uncertainties in rapidly evolving computing technology. A flexible approach is required to choose the optimal representations, algorithms, and testing procedures.

An ideal numerical model is:

1. computationally fast and efficient,
2. in need of only modest computer memory,
3. accurate, both numerically and physically,
4. robust and reliable over a known range of conditions,
5. flexible and easy to modify,
6. quick to produce useful results, and
7. well documented.

Unfortunately, these desirable qualities usually conflict with each other. If an algorithm is very fast, it is seldom the most accurate. If an algorithm is both fast and accurate, its range of applicability is often limited. If an algorithm is valid over a wide range, it may require large amounts of computer memory or be inflexible with respect to changes in boundary conditions. The most accurate algorithms for modeling important physical effects often border on numerical instability, so that computer models using them are not robust and reliable. These constraints and costs are all expressions of a cosmic theorem which says “there is no free lunch.” Making the right choices to optimize the model and

its performance involves assigning relative importance to the qualities listed above. These weights depend on the characteristics of the particular class of problems as well as the abilities and knowledge of those performing the computations.

Consider trade-offs involving computational speed versus computer memory. In fluid simulations, it is possible to recompute the fluid pressure from the equation of state every time it is used during a timestep, or you may decide to compute the pressure once and store it for use elsewhere. The right choice may differ for a two-dimensional or a three-dimensional model, because of the different relative weights on speed and memory for the two cases. Storage is often at a premium in a three-dimensional calculation, although modern, large-memory computers have significantly alleviated this problem. Even when the overall memory is not limiting, the time penalty to communicate additional variables between the separate processors (chips) of a computer may still favor recomputing the pressure each time it is used.

Storage limitations are usually of little concern in a one-dimensional problem, but this also is not a hard and fast rule. Most modern computer processors have a cache to reduce memory access for repeated operations and loops. This cache is usually quite small, but a factor of three or four speed-up is possible by keeping significant portions of the computation in the cache. Information transfer time into and out of the cache, or "memory bandwidth," then becomes a limiting factor, not computing time directly. In this case, it may again pay to store the information rather than recompute it for a three-dimensional model. Indeed, even changing the storage patterns of a large one-dimensional problem could pay big dividends.

Consider some other examples of algorithmic trade-offs. Often one-dimensional algorithms do not generalize easily to two or three dimensions. If the standard problem is three dimensional, a slower, more general algorithm should be used in a preliminary one-dimensional test, keeping in mind that this is the method that will be incorporated in the three-dimensional model.

The choice of algorithm may depend on whether the flows are subsonic or supersonic. The best algorithms for convective transport in flames allow considerably longer timesteps than those required for detonations, even though each timestep requires more computer time. Thus, it might be best to develop the model with the more expensive, shorter-timestep algorithms for shocked flow from the beginning if we ultimately need to model the transition of a flame to a detonation. Finally, if we are interested in solving a set of equations for a large chemical-reaction mechanism, the algorithm we choose depends on whether the calculation will be coupled to fluid dynamics. As discussed in Chapter 5 , the stand-alone solution for chemical kinetics often requires more expensive, more accurate methods. Trade-offs in choosing algorithms are a major part of the discussions throughout this book.

3–3.2. Steps in Constructing a Simulation Model

Before programming a simulation model, major decisions have to be made to

- select a level of modeling commensurate with the problem, computational resources, and the expertise available

- choose a suitably idealized standard case that stresses the most important physical effects and allows subsequent generalizations
- develop a mathematical model that includes the standard case and reasonable deviations

Then program development can begin. The following steps form a check list for constructing a reactive-flow simulation model.

Step 1. Write a control program for the simulation that transfers control to, or *calls*, subprograms to calculate each physical process independently. To begin with, the calls are calls to *dummy* subprograms, which simply return to the main control program. Each dummy subprogram will eventually solve the equations for one of the physical processes being simulated. The control program with the calls to dummy subprograms becomes the outline that is filled in as each dummy routine is replaced by a subroutine that actually performs a calculation. The logic of program control, storing data, restarting a calculation, and diagnostics of results can be developed at this step. An important thrust in scientific computing is to develop a technology for these "skeleton" programs so the organization of multiple-physics and fluid-dynamics modules can be standardized. It is often prudent to bring in generic graphical diagnostic capabilities from the very beginning. These help all subsequent debugging and testing activities.

Step 2. Create a working list of all physical and chemical variables, parameters, control variables, and indices. Determine the computer memory they will require as a function of the dimensions of variable arrays. Physical variables include quantities such as density, momentum, energy, species densities, temperature, and pressure as a function of position. Control variables include logical statements that, for example, turn on processes in the model and control input and output. This list can be used to estimate the memory requirements of the program. The estimate increases as dummy subprograms and subroutines are filled in.

Step 3. Choose a data representation for storing the problem variables. This process lays out the data structure for the numerical implementation. Sometimes different representations may be optimal for different parts of the program. In this case, simple, efficient transformation routines should be written and tested separately. A detailed, schematic diagram of the computational grid should be developed. It should show the cell interface locations, problem geometry, indices, and computational region boundaries. It can also show geometric locations where averaged physical variables are interpolated and indicate what kind of boundary conditions are used. This diagram sets the stage for results presented in tabular or graphical form and will be used repeatedly to explain the model and the results to others.

Step 4. Choose some reasonable initial conditions for a test problem, either for the standard case or for a preliminary benchmark. These data are ideal for testing diagnostics because the values to be printed, plotted, or analyzed are known exactly. It usually pays to have the dummy modules do something artificial but physically reasonable (and with the correct engineering units) to test and exercise the initial suite of diagnostics. As a rule of thumb, most bugs are found in the diagnostic routines, so look there first. They are usually written quickly to get the model working, and thus they are most prone to errors.

Step 5. Write the initializing subroutines that will be called from the control program. These routines fill all the variable and data arrays with the initial conditions for the first standard case. The design criteria here should be simplicity and flexibility. Because the initialization is executed only once during the course of a computation, its efficiency is not as important as its generality. This routine should be data driven and any effort expended to make the data input file self-explanatory will pay off handsomely.

Step 6. Run the program repeatedly with the initializer, the dummies, and the diagnostics until the logical ordering of the processes within the timestep is correct and the diagnostics yield the correct numbers in the correct places with the correct labels. During this phase, it is essential to use known data and to vary these data. Incorrect diagnostic routines can be insidious and produce errors that are interpreted as bugs in the simulation algorithms themselves. This is also a good time to develop intermediate data "dump" facilities to restart the calculation from intermediate simulation data. These restarts should give *exactly* the same numerical results after a given number of cycles as were found running the model straight through without the restart. This will often require saving a number of intermediate control variables in addition to the arrays of primary variables.

Step 7. Begin implementing the physics and chemistry algorithms, each one alone with all of the others turned off. Do not allow two processes to interact before each has been tested alone. This is the stage when many of the algorithm trade-offs discussed above become apparent. Each process simulated should be checked in as many different limits as possible to bound the behavior. Using the simulation to reproduce analytic solutions is the best way to check the numerical model. These solutions give detailed profiles for comparison, so not only the correctness but also the accuracy can be evaluated. Later, when you are using the model for a more difficult problem, you will not know the answers *a priori*. Therefore, a long list of artificial test problems gives confidence in the accuracy and limitations of the model.

Step 8. Test the interactions among the modules by doing calculations in which only two are turned on at one time. When there are many possible interactions, many tests are necessary. Again, limiting cases should be tested and comparisons with theoretical models should be conducted wherever possible. During Steps 7 and 8, it is helpful to modify the model in simple ways, for example, by temporarily changing a diffusion coefficient or gas constant so that comparisons to theoretical test problems are valid, even if the test values are not exactly those used for any of the full simulation cases.

Step 9. Turn everything on . . . and cross your fingers.

3–3.3. Testing Programs and Models

We have recommended choosing an interesting focal problem to help select the computational representation and algorithms, then choosing a specific standard case to implement the model, and then making variations of this standard case to test the model. It is important to use a systematic testing process to evaluate the components of the model before the entire simulation is put together. It can also give answers for direct comparison with either experimental data or analytic theories. When comparing calculations to experimental data, such tests are good ways to exercise modules in the simulation model. When

comparing calculations to analytic theories, it is often possible to predict the results of a theory and then computationally carry the answer into a regime where the theory is no longer valid. This process shows how the computer model works, when you can expect it to stop working, and what the costs are to get reliable answers.

Once the model is constructed, it is necessary to make sure that the physical processes are represented correctly and that the model is as bug free as possible. At this point, knowledge and intuition of the behavior of the important physical processes become important. The remaining material in this chapter is not intended as an organized methodology, but rather is a collection of hints and strong suggestions for developing and testing the model.

Avoid Programming in Dimensionless Numbers

This sounds contrary to everything a student is taught in theory classes. We have had many students (usually laughing) and professors (usually annoyed) bring this controversy to our attention. Indeed, the last ten years has firmly fixed the crucial importance of this *rule* in our minds. Picking a set of standard units, and keeping the program variables in these units, can save a large amount of time in writing, debugging, and interpreting complex reactive-flow simulations with many interacting physical processes. It is inordinately difficult to debug a program when the size of all of the variables is of order one. In practice, most "factor-of-two" errors do not occur, or are much easier to find, if the units in the program relate directly to a real system.

It is also questionable how much can be gained from scaling when there are many interacting processes on many time scales or when the problem is spatially inhomogeneous. The results always can be made nondimensional in the output portion of the program. Working in real units that are relevant to the focal problem helps develop intuition about the system and makes it easier to communicate with engineers and experimentalists. Working exclusively with dimensionless units, although it imparts a patina of academic generality, becomes a crutch. Remember, if you undertake the development and use of a reactive-flow model, it will not be a part-time hobby. Students who graduate without a firm foundation in physical units and a suitably developed intuition for the size, speed, and duration of events, are less capable and less employable than those who have this foundation.

Intermediate Benchmarks

The objective is an accurate, informative simulation of the standard case. To reach this, a number of smaller benchmark calculations, each addressing a physical or numerical question, should be carried out. Sometimes it is useful to choose intermediate benchmarks with dimensions or system parameters that are different from the standard case. For example, when characteristic time and space scales vary widely, a benchmark computation on a closely related problem with a smaller range of scales requires fewer timesteps to run and check. This reduces calculation costs during the development and testing phases.

It is also possible to develop and test the simulation model on problems with lower dimensionality than the standard case. This requires less computer time and memory and the answers can be scrutinized more thoroughly than equivalent two- or three-dimensional problems. In this situation, it is important to ensure that the methods being tested generalize easily and efficiently to higher dimensions. The test problems should be meaningful both

from the point of view of solving the specific problem and testing the computational implementation.

When a benchmark is successfully completed, a copy of the whole computer program that produced it, along with the results it generates, should be filed safely away. This provides a description of the model as it evolves. Hints for documentation are discussed in Section 3–4.

Keep the Computer Program Clean

There is a tendency not to remove parts of computer programs that are no longer used, but to turn them off internally in the model by "commenting" them out or branching around them. Keeping these parts in the program slows down the whole computational process from editing through printing. The superfluous statements can also cause problems if they are accidently turned on or parts are not turned off properly. It is simpler and safer to discard the rejected or incorrect statements, and to reprogram or reinsert them again when they are needed.

Change the Model Slowly and Systematically

It is extremely important not to make too many changes at once when the model is being upgraded or modified. It is better to perform several independent tests that can be compared with each other and with the standard case. Even when new ideas or components seem to be logically independent, they often are not. Thus separate tests not only reduce the overall development time, they also increase confidence in the model. Putting three or four tests into a single run, unless they are serial and truly independent, invites confusion and possibly trouble. Make sure to keep copies of what you did in a log, along with the answers and the versions of the routines that generated them.

Document the Model Internally

Internal model documentation is an important aspect of both changing and developing models. Section 3–4.2 is dedicated to this subject. Everyone has trouble remembering exactly what was done six months or a year ago. Although it may seem like a waste of time to go back and document an apparently working program, you will be thankful that you did. Even if you write the program and do not expect others to use it, you will profit from putting in prolific internal documentation. For example, you often discover special cases that require additional programming when you explain to yourself just what you are doing. You also discover ambiguities in variable names and logical tests that may later cause problems.

Interactive versus Batch Mode

Whether to use interactive or batch mode for computing is a part of the question of how to use your personal and computer time best. Much of the preliminary model construction and testing described above is best done interactively. As the project proceeds, however, there comes a point when the tests take more computer time to complete. Interactive debugging can be unnecessarily time consuming and does not leave you with an unambiguous record to document the progress. Colleagues may also become irate over the amount of high-priority interactive computing required as the calculations get bigger and longer.

After some point, the interactive session is best for editing and submitting different batch tests and for checking the results as they proceed and when they are complete, not for doing the calculations. Debugging calculations for an eventual five-hour simulation requires five or ten minutes to test the changes being considered. In five minutes, four different batch tests can be devised and submitted. Then the next thirty minutes can be used for thinking, writing, or programming while these batch runs are executing.

Modern computing environments are also blurring this distinction. A calculation can be started interactively and then moved to the background in a UNIX session. This usually leaves it at higher priority than a true batch submission but at least allows you to do something else in the same session. Sometimes new computer systems do not have suitable batch scheduling capabilities in the first year or two, and so you will have to work interactively. In other cases, their network connections may initially be too unreliable for effective interactive work.

Maintain a Healthy Skepticism

As a rule of thumb, you should never believe the results of your calculations. Always look for errors, limitations, unnatural constraints, inaccurate inputs, and places where algorithms break down. Nothing works until it is tested, and it probably does not really work even then. This skepticism should help avoid embarrassing situations, such as occur when a neat physical explanation is painstakingly developed for a computational (or experimental) blunder. Our computer center is called the "Miracle Center" to constantly remind us of this. When mass increases by a factor of two every timestep, it is probably *not* a miracle.

3–4. Using Time Efficiently

Developing and testing computer programs, looking at and thinking about the results, and planning the next stage of the project, all take time. We now discuss four aspects of using time efficiently:

1. doing a number of tasks in parallel
2. documenting the entire process
3. incorporating what others do
4. starting over without wasting more time

3–4.1. Partitioning Time among Tasks

Approaching the simulation research in a completely serial way, without having the next and the previous steps in mind, can be costly. This is also true when testing and applying the evolving numerical model. Time is best spent divided among several aspects of the project, some requiring deep thought and others that are mundane. The trick is to find a good use for what would otherwise be dead time. To accomplish this, it helps to do several things in parallel so that progress is always being made on some aspect of the problem. For example, while programs are running or graphical output is being processed, it might be useful to plan the next stages of the project.

Even when the computer is slow, understanding the results takes time and can occur in the background while more mundane tasks are done. A very useful technique is to have selected graphics from an executing simulation, perhaps just a couple of contoured cross-sections or line plots, brought automatically to a window in the corner of your desktop screen. The evolution of the ongoing simulation will generally be slow, but it is likely well matched to your comprehension rate. Furthermore, this dynamic understanding can be acquired while you are doing something else on the same screen in the foreground.

3–4.2. Documenting and Retrieving Results of Simulations

Recording the reasoning involved in making technical decisions, documenting the individual steps in the research, and writing accounts of the work as it proceeds are not universally adopted practices, though most will agree that research should be done this way. Under time pressure, it is easy to "forget" to document exactly how the algorithms are implemented, details in the computer program, or results of the calculations. Improper or sparse documentation can waste a lot of time. Conscientiously documenting the analysis, the algorithms, the programs and subroutines, the methodology for using them, and the computed results can save hours or days in the future at the cost of minutes or hours while you still understand what is going on. Documenting focuses your thoughts and makes it easier to understand what has been done. It helps you spot easier ways to do things early in the development when the improvements will help the most. It is especially important when trying to repeat previous results and to solve follow-on problems that build on current work.

A log, kept faithfully through the model construction and testing stages, keeps track of what worked, where problems were, and what was fixed. In today's scientific research environment, you will naturally tend to keep this project log electronically. It is often useful to keep a handwritten notebook as well, as a kind of index. It can be useful in noting where things are in the electronic files, jotting notes to yourself in scientific meetings, and recording possible articles or books for future reference. Various tests of individual modules and sets of modules can be documented in the electronic log, simply by cutting and pasting appropriate program fragments or segments of files. These should include output files from these tests and a few examples of diagnostic graphics captured from your desktop monitor, labeled with dates, and titled so that they are easily identified. If one algorithm is replaced by another, the changes should be noted and explained. Performing and documenting a comparison immediately may be the only way to capture the reason for a change. If you wait, other things will become more important than making a meaningful comparison. Even if you wish to make a detailed comparison later, you may not be able to because a number of other aspects of the composite model may also have changed in the meantime.

An important part of documentation during the computer model construction and testing phases is documentation in the simulation program itself. This should be done liberally while you still remember what the program is doing. Such documentation could be comments written every few lines into the program to explain what is currently being evaluated or set up. The careful choice of program variable names is at least as important. For example, *amb_press* is easy to interpret as "ambient pressure" for most scientific users, whereas *apre* is relatively cryptic and *standard_ambient_pressure* is too verbose. Strive for a useful

compromise, not a secret shorthand. Documentation can also be added line by line, as appropriate. For example, the Fortran statement,

amb_press = 1.113E + 6 ! (erg/cc) standard ambient pressure

gives the programmer, the interested reader, and the computer necessary information.

The input data and output data from a simulation should also be documented. Each output *dump* file, which contains all of the variables needed to restart the computation at an intermediate timestep, should be labeled internally so that reading the file tells what the data (and the date of execution) are. This labeling should be generated automatically when the model is executed on the computer. Thus an up-to-date record is produced that labels all of the output files. Editing these automatic log files is a useful way to keep an accurate record of what has taken place.

Deciding when to keep something and when to throw it away is difficult. The replacement of paper output and graphical hard copy by files on an electronic storage system has made it possible to keep more worthless output and to forget what is stored more easily. Without a good filing system and a good notebook, it is difficult to find things quickly and reliably. Careful choice of mnemonic names for files and ways to inspect files quickly are extremely useful but no guarantee of success.

Finally, the project notebook should include the standard case and related simulations following it. Output must be labeled with dates and descriptions coupling it to the corresponding input data. A good way to further document results is to generate problem-specific figures and graphs as the problem proceeds. Each figure should be chosen to summarize a certain result, aspect, or concept in the project. For each of these summary figures, an extended figure caption should be written, typically a few hundred words. In addition to having presentations and progress reports ready instantly, the figures and extended captions can be assembled into technical reports. This procedure is similar to using collected internal documentation from the computer program to produce a document describing the computer model. One of the first figures produced in a simulation project should be a schematic figure explaining the computational grid.

3–4.3. Some General Programming Guidelines

The basic simulation model is either developed or acquired early in the course of a project. Just as in laboratory experiments, a significant and sometimes dominant fraction of the technical effort is spent on developing and using new diagnostics, and then analyzing the results they produce. The basic model becomes surrounded by a suite of special programs that quickly becomes more extensive than the original model. These special utilities use the primary simulation variables to create, for example, graphics for interpreting the output or integrated quantities to compare with experiments or theories.

Joining forces with other users of the same or similar computers can reduce the effort of producing and debugging these utility programs. Software obtained this way is seldom exactly what is required, but it can often be changed easily at little cost. Sometimes it is only necessary to write an interface program to transfer data from the numerical model to the diagnostic software. A considerable effort is being expended to do this in these interface programs and special scripting languages and software. The objective is to simplify the

process of performing *and* documenting simulations, but technology often becomes more complex before it is simplified.

If you are working alone and developing software for yourself, you should still proceed as if you are developing the program jointly with collaborators. In all cases a program that implements or diagnoses the simulation must be:

Clear and well documented – It is much easier to understand and modify a clearly written, internally documented program. It is also important in a professional research environment to be able to return to and quickly understand a program written a year or two earlier.

Simple – A simple program is much less prone to error and is more easily modified than one that is convoluted and complicated. Simplicity and clarity also help make the program flexible. Flexibility is not a necessary quality, as it is often best if a program only performs one kind of calculation well. In some cases, however, flexibility is important and highly desirable.

Fast – Making the program fast has economic benefits. It also allows flexibility because some of the features that make a program efficient can be traded off for greater accuracy. Making the program fast also has the benefit of allowing longer calculations with larger grids and better resolution. The quest for performance can rapidly become counterproductive, however, when it detracts from the primary goal of solving a particular reactive-flow problem well enough.

Accurate – The compromise between speed and accuracy has always posed one of the most troublesome and provocative challenges for the scientific programmer. Almost all advances in numerical analysis have come about trying to reach these twin goals. Changes in the basic algorithms will give greater improvements in accuracy and speed than using special numerical tricks or changing programming languages.

Robust – A robust program works adequately over a broad spectrum of input data and has few special cases that it cannot handle. A robust program can be trusted to give correct answers reliably.

Much has been written about structured programming and programming style. Good references on programming style can be found, for example, in McCracken and Salmon (1988), Golub and Ortega (1992), Kheir (1996), Metcalf and Reid (1996), and Etter (1997). Scientific programming is an art, is done in a difficult medium, and requires a special talent. Guidelines are therefore more appropriate than rigidly enforced standards such as might be developed for documentation.

Much has also been written about programming languages. Algol was meant to replace Fortran, PL–1 to replace Algol and Fortran, C to replace PL–1 and Fortran, and object-oriented C^{++} to replace C and Fortran. And there are also ADA, Pascal, Basic, and Cobol. Fortran is still here, having evolved over the last three decades to incorporate those useful new concepts found in other languages that did not conflict with compiling efficient codes. Fortran 90 has much of the additional flexibility provided by C^{++}. To paraphrase a famous quote in scientific computing: "I don't know what language I'll be programming in the year 2000, but it will be called Fortran." Therefore, most of the examples reflect our experience with Fortran.

More on Clarity and Documentation

> Claritymeansinpartreadability.Readabilitycanassumemanyformsand
> shapessothebestsingleguidelinewecancomeupwithistoforcetheprogramtore
> semblewhatmostofusseemorethananyothertypeofvisualcommunication
> Englishprose.Inthiswayitiseasiesttoassimilateprograminfor
> mation.Thereareundoubtedlyavastnumberofwaystoforcetheprogramstate
> mentsintoamuchclearerandmorereadableform.Unfortunatelymanypro
> grammersdothisaboutasmuchasthisrunonbutratherfortranlikepara
> graphdoes.Legibilityisreallyonlyanextensionofthegoldenrule.

Put blanks in the program statements. Put blanks after commas, between variable names, after subroutine names, around + and − signs, and between words. Blank lines make the program neater and more easily read. These are courtesies to the next programmer (most often yourself).

Proper commenting helps to make the program clearer. The comments and the program statements should agree, and the comments should add overview information or interpretation that is not obvious from reading the program statements. The comments should also remind the reader of the overall goal of that particular part of the program, as well as any hidden conditions that it assumes. For easy reading, indent the program statements from the introductory comments, so that the program looks like an outline with the comments as topic headings.

The layout of the program can be almost as useful as the comments in helping to explain what is being done. It is possible to format and structure a program to help the reader find:

a. the structure of program execution – including loops, IF THEN ELSE tests, data structures, and common blocks
b. the data structure – statements that give dimensions to variables, variable declarations, data statements that initialize variables, and assignment statements that change their values
c. the other parts of compound structures – the ends of loops, the labels that are targets of GO TOs, and the format statements that go with READ and WRITE instructions

There are many different concepts of good program layout. Whatever style is chosen, it should make these structures easy to find and scan. Programs are clearer and bugs are easier to find if the program follows a top-to-bottom logical path. Branches and tests should not transfer control to earlier statements unless absolutely necessary. When a back transfer is useful, a comment explaining it is usually in order. Some authors suggest eliminating the GO TO statement altogether, a strong but tenable position.

Do not be afraid to rewrite sections of the program completely. No amount of patching can fix a badly structured program. The time lost to do this is usually saved many times over by avoiding later bugs and confusion, and by. . . finding bugs you did not even know existed.

Break program statements in logical places. Try to use a convenient operation at an outer level to begin each continuation line. Use short constructions, if they suffice, but do not abbreviate comments or variable identifiers. Do not sacrifice readability to save a few lines or spaces. Just as in English, do not mix several ideas into a single line or statement of the program. It is best to keep the program in clear English and simple algebra.

There are some places in programs where it pays to be wordy. It never hurts to use parentheses to avoid ambiguity or to avoid mixed-mode arithmetic that can cause misunderstandings between the programmer and the computer. If a logical expression in the program is hard to understand, it is worth the time to transform it, even if the more legible form is longer.

There are also some places where it pays to be terse. For example, it often pays to avoid temporary variables and unnecessary branches. Avoid IF statements with multiple destinations. A logical expression will often substitute for a conditional branch, will usually compile as efficiently, and can often shorten the program by eliminating the need for an explanatory comment.

Input and output data statements are the most tedious and hence poorly written aspects of programs. It is useful to test all input for plausibility and validity as it is read into the program. Identifying bad input allows the problem to be fixed or the program execution stopped. In particular, if the algorithm has a range of validity, the input should be checked to make sure this range is not exceeded. Print out explanations when the data are suspect. Uniform input formats make data easy to read. The program should echo all the input data in a neat, well-documented format that can be cut out and used to document the simulation.

Debugging is easier when good programming techniques are used. For example, all variables should be initialized before they are used. Constants can be initialized by statements in the program, while variables should be initialized by input or executable statements. When free-form input is used to overwrite default values (for example, the NAMELIST facility in Fortran), both the current value and the default values should be printed in the output and identified there. Otherwise, the user of the program quickly forgets what the default values are and must initialize every value anyway, making the default values worthless. Take care to branch the correct way. Be careful when a loop exits to the same statement from both the side and the bottom. Do not compare floating point numbers solely for equality, since 10.0×0.1 is often not exactly 1.0 in a computer.

Go through the program by hand, using reasonable input numbers. This can be an extremely useful way to find logical and typographical errors. It sometimes works when nothing else does. The trick here is to try to do what the computer has actually been programmed to do, which is not necessarily what you want it to do. We call this "playing computer." A particular advantage of playing computer is that it forces an analysis of what is actually written in the program and exactly what the algorithm does.

After the numerical model is properly understood by both the computer and the programmer and is running correctly, it might be useful to consider ways to speed it up. It may be possible to vectorize or parallelize parts of the program. It may also be possible to find ways to arrange blocks of program to execute more efficiently, such as computing and storing intermediate values rather than recomputing them.

Developing good code structure and good habits for internally documenting codes is as important when you are programming for yourself as when you develop software for others to use. There are, however, additional considerations when another person is going to use the program. What may be obvious to someone can be inscrutable for someone else. Therefore, glossaries of variables, control parameters, subroutine arguments, and logical segments of the program are necessary for a new user and can be extremely helpful for the original author of the program.

3–4.4. Consider Starting Over

In the process of developing, testing, and using the numerical simulation model, changes occur in our perceptions of the problem and thus in what we wish the model to do. The simulations are being performed to extend our understanding of a particular kind of problem. As our understanding evolves, the specification of the job that must be done naturally changes. It is important to be flexible and not to be afraid to backtrack and rethink what you are doing. It is better to change the algorithms or structures as early as possible to save much larger efforts later.

There is another level to the question of starting over. It occurs much later in the life of the numerical model. After being used for a while, a large simulation model becomes a hodgepodge, written partly by the person who originally wanted to solve a problem, partly by those who took over the model and made changes for their own applications, and partly by others unrelated to the project who wrote software that is being adapted and interfaced. The model grows as new pieces are added. Eventually it grows to the point where it is no longer clear, simple, or fast, and there are faster, more accurate, and more robust algorithms to use for modeling the various processes.

There eventually comes a decision point. Do you try to replace old algorithms with new ones and add modification, or do you start completely over and rewrite the program? The answer depends on just how unwieldy the program has become. If it is fairly new, perhaps a few years old, replacing old algorithms and cleaning house may be the best route. But, *never be afraid to start over*. This is a good opportunity to get rid of unused parts of the program, upgrade the algorithms and data structures, and fix the model to use the configuration and special capabilities of the newest available computers properly. The general guideline here is to adapt and clean up until some major change is required, and then to rewrite.

3–5. Programming for Parallel Computers

Since reactive-flow models generally require substantial computational power, it is natural to plan to run them on the biggest, fastest computers conveniently available. Where to run the model is a complex decision on which the success of the project often rests. The choice of computer or computers brings factors outside of your direct control into play, as discussed in Section 3–1.2. The relative difficulty in programming a parallel computer efficiently may be an important drawback but it usually enables freer access to much more computer power than might otherwise be available. Extremely useful discussions of computing can be found, for example, in Fox and colleagues (1988), Ragsdale (1991), Lewis (1993), Reif (1993), Almasi and Gottlieb (1994), Sabot (1995), Wilson (1995), Metcalf and Reid (1996), and Dongarra and colleagues (1998).

3–5.1. Parallel Processing: The Future of Large-Scale Scientific Computing?

In most cases existing programs do not run well on parallel computers, and a rather complete rework is needed to make them work efficiently. Parallel computers are also more difficult to develop new programs for and even to use once the program is fully developed. Nevertheless, there are clear indications that large scientific computers will

continue to become more and more parallel. The underlying reasons for this trend are both economic and technical. The memory and input/output (I/O) subsystems are now a large fraction of the cost of high-performance computers and can be shared among multiple processors. The cost of a very high speed, one-of-a-kind processor is considerably greater than for a lower-speed processor that is being manufactured and sold in large numbers. Therefore, constructing a very large computer out of the same processing and memory chips used in smaller computers is both technically safer and economically advantageous. Given the rapid growth in parallel computers and the steadily improving software for them, it seems very likely that the eventual question is not *whether* to embrace parallel processing, but *when*.

3–5.2. Programming for Parallel Computers

Individual users, of course, have to judge for themselves when to undertake this rite of passage. Once the basic concepts of "parallelization" are learned and have become second nature, developing parallel algorithms and programs becomes considerably easier. In many cases the difficulty stems not so much from the choice of the specific algorithms as from the data structures and the overall implementation of the mathematical model itself. This difficulty is compounded by the fact that parallel computers, even those made in successive generations by the same manufacturer, have a range of architectures. Each of these architectures responds differently to specific attempts at parallelization.

Therefore, we will restrict ourselves to a few general ideas and approaches in the discussion here. Your primary approach should be to find one or two people close to you who know a lot more about the particular parallel computer system you are going to use than you do. These people will be your "gurus" and you are going to ride their coat tails. Be nice to them and help them whenever possible because they can save you untold hours of frustration. The very best situation is to find a guru whose own project has a computational framework or skeleton whose structure you can adopt and then modify to your particular reactive-flow problem.

This skeleton does not necessarily have to be a reactive-flow code, though the closer the problem is to the one you want to solve, the better. A complex-geometry, time-accurate, electromagnetic simulation model, for example, might have a good skeleton for your reactive-flow model. You can adapt the interprocessor communications software to your own needs as you get more proficient. You can start with the data structures your gurus have found most satisfactory and they can probably advise you as to which changes might be a better choice for you. Such skeletons also provide you ready-made input–output facilities to start a run, give you control algorithms for the timestep or corresponding convergence criteria, and diagnostic facilities to check and analyze your computations. You can edit your program into existence efficiently while always having something that executes.

A rather significant transition in viewpoint is necessary to write efficient parallel code where you once produced efficient serial code. You must simultaneously be able to look at the overall structures of the problem you are trying to solve, like the CEO of a company controlling its different operating divisions, and to look at the particular computing

algorithms, like the foreman of a particular assembly line within one of the divisions of the company.

The overview leads to task-level parallelism. Different processors can be doing totally distinct tasks within the overall problem, if these tasks can be identified and effectively separated. The easiest solution seems to be to assign a particular processor the duty of controlling the others. This is usually called "master-slave" programming. The "master" passes out work, in the form of specific computational tasks, and checks on the progress of the "slaves," passing them more work when they are done with what they have. The problem with this approach is that it does not generally scale up to very large problems. Eventually the master processor becomes saturated (bogged down in communications) when there are many slaves. Because the entire flow of the solution passes through the master every timestep, this saturation can occur at rather small problem sizes even with rather powerful processors. While the master is communicating with one of the slaves, a number of the others may be waiting for work. This is called a "load-balancing problem" and we will return to it later.

The macroscopic view is also concerned with the relative merits of bringing the data to the processors or bringing the processors to the data, which is generally called "data control." In general, it is good to minimize communications of data as this activity is generally slower than computation and accomplishes nothing toward advancing the solution. This issue goes hand in hand with the choice of architecture. Parallel computers for scientific use divide generally into two classes, those that employ "shared memory" and those that have "distributed memory." The shared-memory systems typically have a few dozen processors at most and are generally easier to program because the hardware is capable of resolving the conflicting memory references to a significant degree. The distributed memory systems can have hundreds or even thousands of processors, each of which has its own memory. To perform any particular computation, one processor on a distributed memory system has to get its data by communication from another processor.

Without an overall master-slave program structure, each distributed-memory processor has to know where the specific data it needs resides and initiate the required data transfers. This is an example of the microscopic viewpoint. Managing this interprocessor communication is a major part of the difference between programming for distributed-memory parallel processors and for shared-memory multicomputers. Communications programming is extra work for the programmer but is one place where sensible optimization can pay the biggest dividends. To ease the task of performing these data transfers, communications libraries have been developed that include the necessary synchronization calls between processors to allow progression of the model solution without a master processor. Two of these libraries, PVM (Parallel Virtual Machine) and MPI (Message Passing Interface) have achieved some level of universality and may even be used between computers of different architecture connected by a network (Geist et al. 1994; Gropp, Lusk, and Skjelum 1994; Snir et al. 1995).

There are significant pros and cons involved in choosing between shared-memory and distributed-memory systems. The compilers for a shared-memory computer will generally recognize long loops in your program and automatically divide them among the processors assigned to your run. Thus very little extra programming is needed to get a modest amount

of parallel speed-up on such a system. Unfortunately this solution methodology doesn't really scale to very large problems with a large number of processors. Typically this automatic, microscopic "loop-level" parallelism will speed up a calculation by a factor of three or four when four or five processors are used. This is often enough to take real advantage of additional resources when they are available. When twenty processors are applied to such a job, this efficiency usually degrades to below 50 percent.

In contrast, distributed-memory programming requires more work, if for nothing else than writing the communications, but the dividends can be correspondingly large. Sensible programming for a large number of distributed-memory processors does scale. Thus 75 percent to 80 percent efficiency in parallel speed-up is common for problems using hundreds of processors. Careful consideration of the interprocessor communications is also a natural way to ensure load balancing between the distinct processors. Furthermore, distributed-memory programming makes the cost of not properly decomposing the solution very explicit. There is an overhead cost associated with starting up every communication, whether it is between processors on a distributed-memory system or from shared memory to the individual processors on a shared-memory system. Distributed-memory programmers tend to concentrate their messages into a few relatively long messages by restructuring the solution algorithms to allow a great deal of local computation without requiring interprocessor communication. By contrast, this careful thought is seldom given to shared-memory or "global-memory" programming. Many more short messages are generated by the compiler using the programmer's convenient microscopic constructs and this translates to a significant loss of program efficiency. Hidden performance factors of up to ten can be lost this way.

There is another dividend from using the domain decompositions and special data structures associated with minimizing the communications for distributed-memory programming: the program is generally portable across a much wider range of computing systems because it will also run efficiently on shared-memory computers. Often the efficient parcels of computation associated with distributed-memory communications can be made to fit conveniently in the local memory cache of each processor on a shared-memory system. Thus a well-written distributed-memory program can actually be more efficient on a shared-memory computer than programs written in the globally shared-memory paradigm.

Finally, the program must always consider load balancing. A logically convenient approach to task-level decomposition is to have different processes such as chemistry integration, fluid dynamics integration, and evaluation of diffusive transport terms treated by different processor sets. Every timestep, these partial results have to be merged to advance the solution by the time increment Δt. Each of these distinct tasks encompasses a different amount of work so it is possible that the tasks will not finish at the same time. Those processors that finish first will be idle, one instance of the load-balancing problem. Load balancing is a problem that should be approached both macroscopically through the problem structure and microscopically by the individual processors. If the chemistry for one million grid points takes much longer than the fluid dynamics, for example, it makes sense to assign proportionally more processors to the chemistry. This is structural load balancing.

It often happens that the relative amount of work between different components of a solution changes in time. Perhaps the chemistry integrations in the automobile refueling

problem discussed earlier have no work because ignition has not taken place. A short time later, when the flame undergoes a transition to a detonation, chemistry integrations may become the most expensive and intensive process being computed, perhaps as much as a factor of ten more expensive than convection. It would then make sense to reapportion the processors between the two tasks accordingly. There are a number of ways to do this using a master processor to monitor the work done by the other processors and decide how to reapportion the work appropriately. Such a solution may be quite appropriate for a shared-memory application. When the number of processors and the data to be reapportioned are large, this load balancing can become a major bottleneck unless each processor takes care of this activity for itself. Microscopic, ("local") strategies for doing this load balancing can be considerably more complex than the centralized master-slave approach and may even depend on the particular computer architecture being used.

In general you should not allow this optimization or parallel-implementation effort to consume you. It is all too easy to get diverted from the main goal of getting accurate answers to specific problems that you have carefully posed for simulation. Aim your efforts at reasonable efficiency for minimal cost of your time. A factor of ten inefficiency is probably intolerable and can probably be removed relatively easily. On the other hand the last factor of two overall inefficiency in your program may take a complete rewrite of the code, may be hidden behind other problems, or may simply not be worth your effort to solve. Remember that in two or three years you will be running on computers that are at least twice as fast. Instead of chasing efficiency for those years, you can be solving real problems and publishing papers about them. The *real* cost for a factor of two inefficiency is only about 20 percent reduced spatial resolution, that is, 1.2^4.

REFERENCES

Almasi, G.S., and A. Gottlieb. 1994. *Highly parallel computing*. 2nd ed. New York: Benjamin/Cummings.

Dongarra, J.J., I.S. Duff, D.C. Sorensen, and H.A. van der Vorst. 1998. *Numerical linear algebra for high-performance computers*. Philadelphia: SIAM.

Etter, D.M. 1997. *Structured Fortran 77 for engineers and scientists*. 5th ed. Menlo Park, Calif.: Addison-Wesley.

Fox, G., M. Johnson, G. Lyzenga, S. Otto, J. Salmon, and D. Wallace. 1988. *Solving problems in concurrent processors. Vol. 1, General techniques and regular problems*. Englewood Cliffs, N.J.: Prentice Hall.

Geist, A., A. Beguelin, J. Dongarra, W. Jiang, R. Manchek, V. Sunderam. 1994. *PVM: Parallel Virtual Machine, A users' guide and tutorial for networked parallel computing*. Cambridge, Mass.: MIT Press.

Golub, G.H., and J.M. Ortega. 1992. *Scientific computing and differential equations: An introduction to numerical methods*. rev. ed. San Diego: Academic.

Gropp, W., E. Lusk, and A. Skjelum. 1994. *Using MPI: Portable parallel programming with the message-passing interface*. Cambridge, Mass.: MIT Press.

Kheir, N.A. 1996. *Systems modeling and computer simulation*. 2nd ed. New York: Marcel Dekker.

Landau, R.H., and P.J. Fink, Jr. 1993. *A scientist's and engineer's guide to workstations and supercomputers: Coping with UNIX, RISC, vectors, and programming*. New York: Wiley.

Lewis, T.G. 1993. *Foundations of parallel programming: A machine-independent approach*. Washington, D.C.: IEEE.

McCracken, D.D., and W.I. Salmon. 1988. *Computing for engineers and scientists with Fortran 77*. 2nd ed. New York: Wiley.

Metcalf, M., and Reid, J.K. 1996. *Fortran 90/95 explained*. New York: Oxford.

Ragsdale, S. 1991. *Parallel programming*. New York: McGraw Hill.
Reif, J.H. 1993. *Synthesis of parallel algorithms*. San Mateo, Calif.: Kaufmann.
Sabot, G.W. 1995. *High performance computing: Problem solving with parallel and vector architectures*. New York: Addison-Wesley.
Snir, M., S.W. Otto, S. Huss-Lederman, S.W. Walker, and J. Dongarra. 1995. *MPI: The complete reference*. Cambridge, Mass.: MIT Press.
Wilson, G.V. 1995. *Practical parallel programming*. Cambridge, Mass.: MIT Press.

4

Some General Numerical Considerations

This chapter presents and analyzes the properties of the simplest finite-difference methods for simulating four of the main physical processes in reactive flows: chemical reactions, diffusion, convection, and wave motion. The material presented is an overview and short course on solving idealized forms of the equations representing these processes. The discussion highlights the features and weaknesses of these solution methods and brings out numerical difficulties that reappear in solutions of the complete set of reactive-flow conservation equations. Throughout the presentation, we list and describe the major computational and algorithmic trade-offs that arise in simulating each process separately.

The material presented here introduces the more advanced solution techniques described in Chapters 5 through 9. Chapter 11 deals with techniques for solving the coupled set of equations that forms the reactive Navier-Stokes equations discussed in Chapter 2. In particular, Chapter 11 shows how the disparate time and space scales of each type of process can be used to determine a reasonable overall timestep for the computation. The choice of the numerical boundary conditions that are so crucial for correctly defining the physical problem, are discussed in Chapters 5 through 9. Sections 10–1 and 10–2 are devoted to issues of selecting boundary conditions for the reactive-flow equations.

Table 4.1 shows the mathematical representations discussed in this chapter and indicates where the numerical solutions for more complex forms of these equations are discussed elsewhere in this book. There are many references on numerical methods and scientific computation for science and engineering that cover material complementary to this chapter. Many of these are listed in the bibliography given in the Prologue. In particular, we recommend the books by Potter (1973), Roache (1982), MacKeown and Newman (1987), Koonin and Meredith (1990), De Jong (1991), Nakamura (1991), Hoffman (1992), Thompson (1992), Wong (1992), DeVries (1994), Garcia (1994), Anderson (1995), Gould and Tobochnik (1996), Press and colleagues (1996a,b), and Tannehill, Anderson, and Pletcher (1997).

Table 4.1. Terms and Corresponding Processes in the Reactive-Flow Equations

Equation	Physical Processes	Where They Are Discussed
$\frac{\partial \rho}{\partial t} = \gamma \rho + S$	Local processes: Source Sink Coupling Chemical reactions	Section 4–2; Chapters 1 and 5
$\frac{\partial \rho}{\partial t} = D \frac{\partial^2 \rho}{\partial x^2}$	Diffusive processes: Molecular diffusion Thermal conduction Thermal diffusion Radiation transport	Section 4–3; Chapters 7 and 13
$\frac{\partial \rho}{\partial t} = - \frac{\partial (\rho v)}{\partial x}$	Convective processes: Advection Compression Rotation Radiation transport	Section 4–4; Chapters 8–10, 13
$\frac{\partial^2 \rho}{\partial t^2} = v_w^2 \frac{\partial^2 \rho}{\partial x^2}$	Waves and oscillations: Sound waves Gravity waves Other oscillations	Section 4–5; Chapter 10

4–1. Introduction to Finite Differences

4–1.1. Discretizing Time and Space

Conventional computers have finite-sized memories segmented into bytes and words of data that may be treated as floating-point numbers. This structure forces us to represent the continuous fluid variables in the computational domain using a finite number of discrete real values. Typically the spatial domain is broken up into *computational cells*, also called *fluid elements* or *zones*. The cells are the volumes defined by a lattice of points, often called the *computational mesh*, the *grid*, or the *cell interfaces*. Usually the values of the physical variables associated with these cells are evaluated at a specific time. Because the computer representation uses discrete numbers rather than continuous variables, the resolution in time is also quantized into discrete intervals. These intervals, called *timesteps*, provide convenient increments over which to advance the numerical solution.

We want to compute solutions for a continuous dependent variable $\rho(x, t)$, where x is a spatial variable and t represents time. The function $\rho(x, t)$ will be approximated by updating the discrete cell values at the times defined by the sequence of discrete timesteps. The variable name ρ is chosen to remind the reader that, in general, our objective is to simulate numerically the properties of a spatially varying density function, such as mass density, momentum density, energy density, or chemical species concentration. The variables ρ and x can be scalar or vector.

Throughout this book, the subscript j indicates the jth computational cell. A sequence of adjacent computational cells is denoted by the indices $\ldots, j-1, j, j+1, \ldots$, and the interfaces separating these cells are labeled $\ldots, j-\frac{1}{2}, j+\frac{1}{2}, \ldots$. The quantity ρ_j^n is the cell-centered quantity ρ at location x_j at time t^n. This quantity ρ_j is the cell value of the continuous variable ρ, typically the average of $\rho(x,t)$ over the computational cell. The coordinate x_j is the location of cell j. The values $\{x_{j+\frac{1}{2}}\}$ label the cell-interface locations between the corresponding cell locations x_j and x_{j+1}. The cell width is given by

$$\Delta x_j = x_{j+\frac{1}{2}} - x_{j-\frac{1}{2}}. \tag{4–1.1}$$

In this chapter, the index j is generally used instead of i to avoid confusion between the cell index and the imaginary number $i = \sqrt{-1}$ that appears in Fourier expansions used to analyze the finite-difference algorithms. We also use k (units of cm^{-1}) to represent the wavenumber of a Fourier mode with spatial dependence $e^{i\mathbf{k}\cdot\mathbf{x}}$. In multidimensional applications, the cells are often labeled i, j, k to represent three spatial directions. Because we rarely use Fourier analysis in multidimensional contexts in this book, there should be no confusion.

The superscript n, for example ρ^n, indicates a particular variable evaluated at the time t^n. Thus a sequence of times are denoted by superscripts $\ldots\, n-1, n, n+1 \ldots$. Superscripts $n-\frac{1}{2}$ or $n+\frac{1}{2}$ indicate times halfway through the timestep. When we write down algorithms that advance an equation by a timestep, we sometimes use the superscript o to denote the value of the variable at the "old" timestep, $n-1$, and then n denotes the "new" timestep. The timestep is usually denoted Δt. In the more general algorithms described in later chapters, Δt can vary with time, so that

$$\Delta t^n \equiv t^n - t^{n-1} = t^n - t^o. \tag{4–1.2}$$

Often, as in Chapter 5, the symbol h is adopted to denote the integration interval. This notation is common in analyzing ordinary differential equations where h can represent an incremental step in any independent variable.

There are a number of ways to interpret the discretized variables in terms of the continuous physical quantities being approximated. Consider x_j, the location of cell j. Depending on how we choose to interpret x_j, it could be

- the location of the cell center,
- the location of the cell center-of-mass, or
- a particular point in the cell where ρ_j happens to be known.

Consider representations for ρ_j^n, the cell value of ρ at x_j. This can be

- the value of $\rho(x,t)$ at a particular position x_j and at a time t^n,
- a characteristic value of $\rho(x,t)$ near the specific location x_j at time t^n,
- an average of the continuous solution in the volume enclosed by the computational cell interfaces at a specific time t^n,
- the average value in time of ρ between $t^{n-\frac{1}{2}}$ and $t^{n+\frac{1}{2}}$ in the cell x_j,

- the average in space and time, or
- the coefficients of a set of basis functions in a linear expansion of the dependent variable $\rho(x, t)$.

Each of these representations has different properties, approximates some situations better than others, and generally gives rise to different solution algorithms.

In each of these representations, there is only a finite number of discrete values in the representation, and each value is only specified to finite precision. The result is uncertainty in the computation arising from the discretization. The source of the uncertainty is the missing information about the detailed solution structure within the discrete cells and timesteps. All approaches that use a finite number of values to represent a continuous profile have this problem. The freedom associated with choosing a representation is considered more extensively in Chapter 6.

Consider a continuous independent variable z defined in the bounded region

$$z_1 \leq z \leq z_2. \tag{4–1.3}$$

Here z represents either a space or a time variable. Figure 4.1 shows two possible profiles of $\rho(z)$, each of which has the same average in the computational cells. This figure shows that including more values in the sequence $\{\rho_j\}$ gives a better representation of the function. When fewer points are used, more information about ρ is lost. By choosing a finite, discrete representation of ρ, we only describe the long-wavelength properties of ρ. If z represents time, choosing a particular timestep is the same as choosing a filter that wipes out high-frequency structure in the solution. If z is a space variable, we effectively filter out the short wavelengths by choosing a particular spatial grid size.

The selection of an appropriate timestep and computational cell size are important decisions in performing simulations. This chapter shows that each type of process in the reactive Navier-Stokes equations may require a specific temporal and spatial resolution. Chapters 11, 12, and 13 show that the combined effects of multiple processes may add additional constraints on the resolution required.

4–1.2. Advancing Equations in Time and Space

The elementary definition of the derivative of a continuous variable ρ with respect to z is

$$\begin{aligned}\frac{d\rho(z)}{dz} &= \lim_{\Delta z \to 0} \frac{\rho(z + \Delta z) - \rho(z)}{\Delta z} \\ &= \lim_{\Delta z \to 0} \frac{\rho(z) - \rho(z - \Delta z)}{\Delta z}.\end{aligned} \tag{4–1.4}$$

The calculus of ordinary and partial differential equations considers the limit, as indicated in equation (4–1.4), as the interval Δz goes to zero. In finite-difference expressions for the derivative, the limit cannot be taken because there is no continuum of values on which to shrink the increment to zero. The step size Δz must remain finite. The derivative is approximated on the computational mesh using the discrete values $\{\rho_j\}$ and $\{\Delta z_j\}$. For

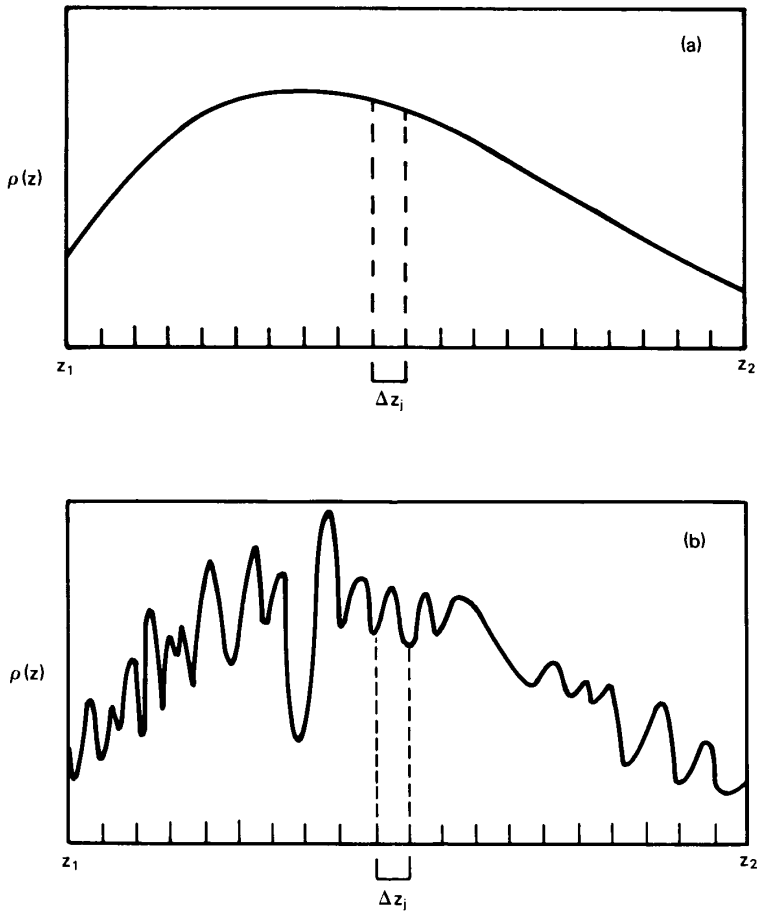

Figure 4.1. Two functions $\rho(z)$. In (a), ρ varies smoothly and does not change much in the discretization interval Δz_j. In (b), ρ has substantial structure in this interval and the discrete values carry relatively little information.

example, one finite-difference approximation to the derivative, the first-order forward difference, is

$$\lim_{\Delta z \to 0} \frac{\rho(z + \Delta z) - \rho(z)}{\Delta z} \approx \frac{\rho_{j+1} - \rho_j}{z_{j+1} - z_j}. \tag{4–1.5}$$

It is usually assumed that a numerical solution of some problem approaches the exact answer as the values of $\{\Delta z_j\}$ become smaller.

Table 4.2 lists several of the simpler finite-difference forms for derivatives that can be used for the terms in Table 4.1. When the solutions are smooth, it is generally true that a higher-order solution means a more accurate solution. Increasing the order of accuracy means increasing the number of values of the variables included in the finite-difference formulas. For example, the algorithms for the first-order derivatives involve ρ_j and either ρ_{j+1} or ρ_{j-1}. For the algorithm to be second order, ρ_{j+1} and ρ_{j-1} are included, and so on. Thus "higher order" usually means more computational work and a wider footprint on the computational grid. When the solutions are not smooth and have appreciable unresolved

Table 4.2. Finite-Difference Forms for Derivatives

Derivative	Form	Order of Accuracy
$\frac{\partial \rho}{\partial z}$	$\frac{\rho_{j+1} - \rho_j}{\Delta z}$	First (forward difference)
	$\frac{\rho_j - \rho_{j-1}}{\Delta z}$	First (backward difference)
	$\frac{\rho_{j+1} - \rho_{j-1}}{2\,\Delta z}$	Second (centered difference)
$\frac{\partial^2 \rho}{\partial z^2}$	$\frac{\rho_{j+1} - 2\rho_j + \rho_{j-1}}{\Delta z^2}$	Second (centered difference)

structure, using high-order difference methods can actually be less accurate than lower-order methods because approximations to higher-order derivatives require information from further away on the grid.

This chapter emphasizes *finite-difference methods* (or *finite-volume methods*), constructed using formulas similar to those in Table 4.2. Other generic types of approaches, such as finite-element and spectral methods, are discussed in later chapters.

4–1.3. Qualitative Properties of Numerical Algorithms

Each of the four processes in Table 4.1 has important qualitative physical properties stemming from the form of the equations being solved. These qualitative properties, listed in Table 4.3, should be reflected in numerical solutions. Much is gained by choosing numerical algorithms that do so. The first four qualitative physical properties (conservation, causality, positivity, and reversibility) are properties of the physical system and of the continuum mathematical representation. The fifth property, accuracy, is a requirement of the quantitative solution of the computational model equations, not of the physical system.

First consider *conservation*. *Conservation laws* are expressed as integrals over continuous fluid quantities. These conservation integrals should remain constant, regardless of the details of the fluctuations and flow phenomena occurring. Usually a conservation law equates the rate of change of the integral throughout a finite volume with the integral of

Table 4.3. Desirable Properties of Numerical Algorithms

Property	Comment
Conservation	Applies to mass, momentum, energy, species number densities, and other conserved quantities
Causality	Due to finite propagation speed, or finite time for information transmission
Positivity	Important for all but wave processes
Reversibility	Important for waves and convection
Accuracy	Algorithm must give quantitatively correct answers, must converge, and must be stable

a flux quantity passing through the surface of that volume. The terms in Table 4.1 are all expressions of such *conservation laws*. The conservation laws can be written in this *flux conservation* form because they reflect local properties of the system – they are derived as integrals of the partial differential equations.

For example, suppose ρ represents a mass density. Local conservation of mass means that the density ρ changes due to the flow of material into and out of the surfaces surrounding a given volume of space. Because any flux leaving one cell through an interface enters an adjacent cell, this formulation conserves ρ globally as well as locally.

Local conservation laws can be used to formulate finite-volume algorithms that conserve sums of the discrete simulated physical quantities. These sums are finite-volume analogues of the conservation integrals. Demanding that a numerical algorithm conserve a specified quantity both locally and globally introduces a constraint that forces the algorithm to represent an important aspect of the physical laws correctly. When conservation laws are not enforced, truncation and round-off errors can grow without bound, making even very simple systems behave in a bizarre manner. Many algorithms discussed in this and later chapters are conservative. If we give up strict conservation in an algorithm, we should demand something else in return. For example, Section 4–2 discusses asymptotic algorithms that do not strictly conserve, but that produce accurate answers for very long timesteps. When a conservation law of the physical system is not built directly into the solution algorithm, how well the algorithm actually conserves the specified quantity becomes a measure of its accuracy.

Physical constraints limit the rate at which material and energy can be transmitted in a physical system. Material or energy going from one location to another must pass through the intermediate locations. This statement of *causality* has important implications for the choice of the solution method. Implicit algorithms, discussed throughout this chapter, tie the solution at one location in space directly to that at far distant locations. Thus they transmit numerical information across distances of many computational cells each timestep. Keep in mind that there can sometimes be problems because this numerical transmission speed is too fast. For example, the material in front of a gas-dynamic shock should have no forewarning that a shock is coming. Therefore, implicit algorithms tend to generate unphysical precursors of an advancing shock in the undisturbed, unshocked fluid.

Positivity means that quantities such as the mass density, which are initially positive everywhere, cannot become negative through the action of the fluid convection and diffusion algorithms. Because a simple continuity equation is linear in the convected variable, superposition of solutions should apply. This means that a constant can be multiplied by a solution and it is still a solution. Thus positivity also means that no new maxima or minima can be generated by convection alone. The somewhat more general term *monotonicity* has come to be used essentially synonymously with positivity, and both terms will be used in this book. This property holds for any quantity described by the convection or diffusion equations. Physical quantities described by wave equations or quantities driven by additional source terms, such as pressure gradients acting on the components of the momentum equation, can change sign even though the underlying convection, when considered by itself, preserves positivity. This semantic confusion has led many people to erroneously believe that an entire solution will be everywhere positive (or at least monotone)

because the algorithm used to represent $\mathbf{v} \cdot \nabla\rho$ generates no spurious oscillations. As shown in Chapters 8 and 9, solutions of the convective transport equation, the third term in Table 4.1, are greatly improved when positivity and conservation are enforced in the solution algorithm. The results of enforcing these properties are the monotone algorithms discussed extensively in Chapter 8.

An equation has the property of *reversibility* if it is invariant under the transformation $t \rightarrow -t$ and $\mathbf{v} \rightarrow -\mathbf{v}$. It is important to preserve reversibility for waves and convection, but not really relevant for diffusion or chemical kinetics. Computationally, reversibility means that if, at time t^n, the calculation is stopped and the timestep is made negative, the solution retraces its evolution back to time t^o. Reversibility is not easy to achieve in practice, even when an algorithm is designed to be reversible, because of nonlinear terms in the equations, a moving grid, or other complicating effects. Therefore, reversibility is usually implemented only approximately.

Accuracy involves computer precision, numerical convergence, and algorithm stability. An algorithm converges when successively reducing Δx and Δt produces successively more accurate answers. A method need not always be convergent to be useful. Asymptotic methods, used near equilibrium or for the solution of stiff equations, are examples of this. Asymptotic numerical approximations do not necessarily get better as the step size is made smaller. These methods are discussed in Section 4–2 and in Chapter 5. An algorithm is stable if a small error at any stage in the calculation produces a smaller cumulative error. All useful numerical algorithms must be stable, at least in the regime where they are used.

4–1.4. Approaches to Analyzing Accuracy

Accuracy criteria quantify how much information is lost when a continuous function is represented by a discrete set of values. A practical approach to determining accuracy is to choose a simple, theoretically soluble linear problem, usually one that has constant coefficients. We then compare the exact mathematical solution to the solution given by the particular finite-difference algorithm being used.

Comparisons of the exact solution and the finite-difference solutions for ordinary differential equations are usually done by comparing coefficients of the successively higher-order terms in power series expansions of the two solutions. For example, assume that ρ is an exponential:

$$\rho(z) = e^{\gamma z} = 1 + (\gamma z) + \frac{(\gamma z)^2}{2} + \frac{(\gamma z)^3}{6} + \cdots, \tag{4–1.6}$$

and the expansion for $\rho(z)$ obtained from a finite-difference algorithm can be written

$$\rho(j\Delta z) = 1 + (\gamma j\ \Delta z) + \frac{(\gamma j \Delta z)^2}{2} + \frac{(\gamma j \Delta z)^3}{4} + \cdots. \tag{4–1.7}$$

This finite-difference solution is said to be *accurate to second order* in Δz and has a third-order error $(\gamma j \Delta z)^3/12$, the difference of the two approximate solutions.

Because finite-difference algorithms for the nonlocal processes in Table 4.1 involve spatial as well as temporal derivatives, they are often evaluated by comparing the exact analytic solution for a single Fourier harmonic with the same Fourier harmonic found by

using a particular finite-difference form. When the problem is linear and the coefficients are constant, each harmonic of the system can be treated independently and analyzed as a separate problem. This allows the same analytical techniques to be used for the partial differential problem as for ordinary differential equations.

The infinite discrete Fourier expansion of the continuous function $\rho(z)$ with periodic boundary conditions is given by

$$\rho(z) = \sum_{k=-\infty}^{\infty} \rho(k) e^{i2\pi kz/(z_2-z_1)}, \tag{4–1.8}$$

where k is here a dimensionless integer. The discrete Fourier transform coefficients $\rho(k)$ used in equation (4–1.8) are defined by the Fourier integral

$$\rho(k) \equiv \frac{1}{z_2 - z_1} \int_{z_1}^{z_2} \rho(z) e^{-i2\pi kz/(z_2-z_1)}\, dz\,. \tag{4–1.9}$$

By expanding the discrete density values on a computational grid as a finite Fourier series for $\{\rho_j\}$, we obtain

$$\rho_j = \sum_{k=1}^{J} \rho(k) e^{-i2\pi kj/J}, \tag{4–1.10}$$

where the discrete Fourier transform coefficients $\rho(k)$ are now defined by a sum rather than an integral,

$$\rho(k) \equiv \frac{1}{J} \sum_{j=1}^{J} \rho_j e^{-i2\pi kj/J}. \tag{4–1.11}$$

As seen from equation (4–1.10), only a finite number of discrete wavelengths can be represented on the finite mesh. The representation $\{\rho_j\}$ is only a long-wavelength approximation to the continuous function $\rho(z)$. There is a cut-off wavelength below which no structure in the solution can be resolved. Analyses of this type are carried out in this and subsequent chapters to indicate the restrictions of the finite-difference approximations for different wavelength phenomena.

When we use Fourier series to analyze a particular algorithm, it is useful to define an *amplification factor* for each wavenumber k in the solution,

$$A(k) \equiv \frac{\rho^n(k)}{\rho^{n-1}(k)}. \tag{4–1.12}$$

The amplification factor is the ratio of the kth Fourier harmonic at the new timestep n to its value at the previous timestep. We show in Section 4–3.4 that $A(k)$ measures how much the amplitude of mode k decays (or grows) during a timestep. To represent smoothing of a gradient, $|A|$ should be less than one. In general, $A(k)$ is a complex number that can be written as

$$A(k) = A_R(k) + i A_I(k) = |A(k)| e^{i\varphi(k)}, \tag{4–1.13}$$

where $\varphi(k)$ is the *phase shift* arising during a single timestep for mode k.

Individual Fourier harmonics are convenient test functions because the discrete Fourier series expansions can be manipulated analytically almost as easily as the continuous Fourier integrals. Thus it is relatively straightforward to see the correspondence between the continuous variable and the discretized approximation. This correspondence is convenient, but it can also be misleading. Because of the global extent of the sinusoidal basis functions in a Fourier expansion, problems often arise in trying to ensure physical causality and positivity. For example, fluid upstream from a shock in a gas is physically unable to respond until the shock actually arrives because the shock travels faster than any characteristic speed in the fluid. In usual Fourier treatments of shocks, numerical precursors arrive ahead of the shock.

4–1.5. Types of Finite-Difference Errors

The worst errors of all are programming errors. Many hours of numerical analysis and reams of paper are often expended when a programming bug is misinterpreted as an accuracy or stability problem endemic to the algorithm. To avoid the confusion of blaming a flawed solution on the algorithm rather than the program, it is important to be able to recognize the types of computational errors that occur in algorithms and what can be done to minimize them. The four types of errors that occur as a result of using finite-difference algorithms are listed in Table 4.4 along with the source of each error and the desirable solution properties that may be affected.

Local errors occur in all algorithms for any of the terms in Table 4.1. They originate from the finite precision of the computer and from hardware errors. They also arise as local discretization errors of the time derivative. Discussions of these types of errors come up frequently throughout this book.

Table 4.4. Four Types of Errors on a Grid

Type of Error	Properties Affected	Sources of Error
Local	Conservation Reversibility Positivity Accuracy	Precision (round-off) Truncation (time derivative) Hardware errors

Amplitude	Conservation
	Reversibility
	Positivity
	Causality
	Accuracy
		Truncation (space derivative)
		  Numerical instability
		  Numerical diffusion
		Precision

Phase	Causality Positivity Accuracy	Truncation (space derivative) Numerical dispersion Precision
Gibbs	Causality Positivity Accuracy	Finite grid size Grid uncertainty Precision

When spatial derivatives are present, errors appear in the phase and amplitude of the amplification factor, $A(k)$, for each Fourier mode. Amplitude errors in convection and wave equations occur when $|A| \neq 1$. When $|A| < 1$, the result is numerical diffusion, or numerical damping, due to too much smoothing. When $|A| > 1$, the result is numerical instability due to too much steepening. Amplitude errors in the diffusion, convection, and propagation terms affect all of the desirable properties in Table 4.4 and are illustrated in Figure 4.2a. With $t_1 < t_2 < t_3$ in the figure, an initially localized quantity spreads out and its peak value and gradients decrease in time. A numerical algorithm in which the amplitudes of the short-wavelength modes decay faster than the larger wavelengths is often

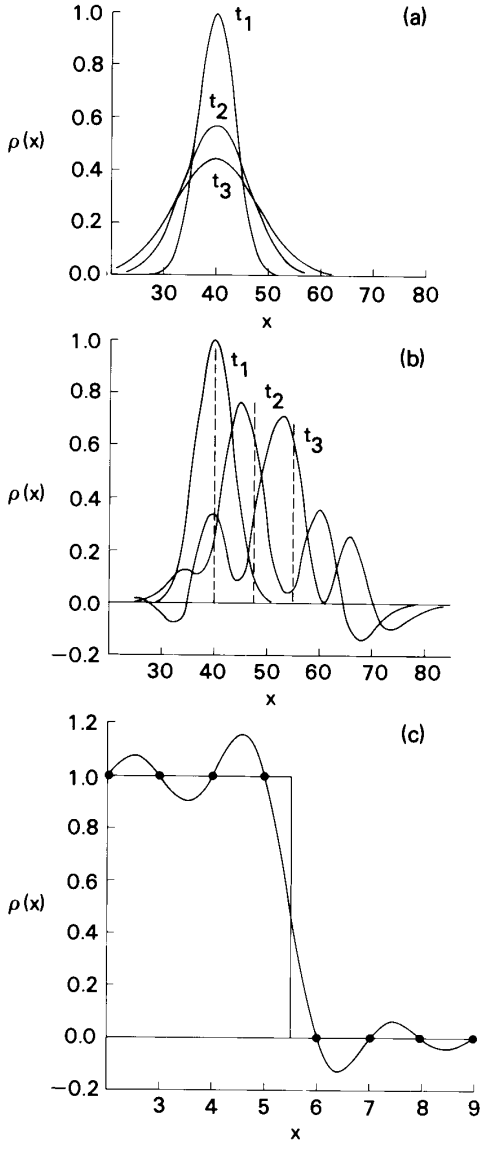

Figure 4.2. Schematic showing three types of numerical errors: (a) Amplitude errors. When $t_1 > t_2 > t_3$, the function ρ should not change in time from its shape at t_1. Numerical diffusion causes ρ to spread. When $t_3 > t_2 > t_1$, ρ should not change from its shape at t_3, but numerical instability amplifies it. (b) Phase (dispersion) errors, which cause the solution to deviate from the profile indicated by the dashed line. (c) Gibbs errors (numerical grid uncertainty) caused by using a finite representation of a continuous function.

said to suffer *numerical diffusion*. This can be thought of as nonphysical damping. In the reverse case, suggested by taking the curves in the sequence $t_1 > t_2 > t_3$, the system is numerically unstable.

Dispersion is a physical phenomenon in which different wavelengths travel at different speeds in a medium. For example, a prism breaks white light into a rainbow through dispersion. Many types of waves are dispersive, but convection should not be. Any dispersion arising from discrete approximations to the spatial derivatives is a numerical error, called *phase error* or *numerical dispersion*. These errors occur in convection and wave equations, but not in local or diffusion equations. Phase errors can adversely affect causality, positivity, and accuracy. The spurious ripples they introduce are illustrated in Figure 4.2b.

A localized quantity of material in a convective flow or a localized acoustic pulse should travel coherently for a long distance. Though the profile is composed of many modes, the phases of all of these modes, in the absence of numerical dispersion, are coupled together to keep the peaks and gradients in the profile well defined and sharp. Numerical dispersion, like numerical diffusion, usually affects the short wavelengths most severely. In the shortest wavelength that can be represented (about two cells), phase changes appear as amplitude errors and thus these wavelengths cannot even propagate. Sharp gradients will develop a train of ripples, as shown in Figure 4.2b for a steep density profile moving to the right.

Phase errors do not always appear as numerical dispersion. If all modes have the same phase error, the wave or the localized profile simply moves at the wrong speed. No dispersion is involved. In numerical diffusion, the errors are limited in size because short-wavelength modes disappear. Unlike numerical diffusion, however, numerical dispersion is not self-limiting. Dispersion errors can increase linearly with time as the phase maxima of different modes move continually further apart. Thus phase errors are the most insidious type of error. Because the amplitudes of the individual modes stay bounded, there is no clearly identifiable point beyond which the solution becomes unsatisfactory.

The fourth type of finite-difference error, the *Gibbs phenomenon* or *Gibbs error*, occurs in wave and convection equations. Gibbs errors look very much like dispersion errors (compare Figures 4.2b and 4.2c), but their origin is different. Gibbs errors occur because the higher harmonics of the continuous solution are not included in the numerical representation. Gibbs errors originate in the very short-wavelength harmonics, shorter than two computational cells, that are not included in the numerical representation. They arise from representing a continuous function on a discrete grid and are *not* due to a particular algorithm. Thus Gibbs errors cannot be eliminated by improving the phase and amplitude properties of the numerical algorithm. They set a bound on the accuracy possible in discrete approximations and can be reduced only by improving the resolution in an Eulerian representation. By moving the nodes in a Lagrangian manner, the Gibbs errors are suppressed at the cost of including additional degrees of freedom.

Gibbs errors appear in calculations as undershoots and overshoots that are largest near steep gradients and discontinuities. These Gibbs oscillations become apparent as soon as a steep profile moves a fraction of a cell from its initial location. Sometimes it appears as if they are not present, and sometimes they seem to disappear temporarily in a calculation. One of the worst aspects of Gibbs errors is the bound they place on the

achievable accuracy of discrete solution methods for convection. This limitation arises from the intrinsic uncertainty of the representation rather than from the properties of the solution algorithm.

Figure 4.2c shows the Gibbs error for a calculation of the translation of a scalar density discontinuity whose height is $\rho = 1$ and whose width is 20 cells. This "square wave" is convected by a method which has no amplitude or phase errors, so only the Gibbs error remains. An example of such a method is based on the spectral representation discussed in Chapters 6 and 8. The dots are the known density values at the grid points. Using the lowest 100 Fourier harmonics, the solid curve is synthesized as the smoothest continuous curve passing through all the known values. When the profile moves a half cell, the Gibbs errors appear as wiggles between the grid points. The resulting unphysical fluctuations are typically of order 10 to 15 percent of the height of the discontinuity. As the grid resolution increases, the affected region shrinks laterally, but the amplitude of the "wiggles" is unchanged.

4–2. Local Processes and Chemical Kinetics

4–2.1. Explicit and Implicit Solutions

The first type of process in Table 4.1 is described by equations of the form

$$\frac{\partial \rho}{\partial t} = \gamma \rho + S, \tag{4–2.1}$$

where ρ is one of the fluid variables such as the mass density or the number density of a chemical species. This simple ordinary differential equation represents spatially localized processes such as local mass, momentum, or energy source and sink terms that may be a function of position as well as time,

$$S(\mathbf{x}, t). \tag{4–2.2}$$

Local processes can also have the linear form described in the first term on the right hand side of equation (4–2.1),

$$\gamma(\mathbf{x}, t)\rho(\mathbf{x}, t). \tag{4–2.3}$$

Here γ is an exponential growth or damping coefficient when it is positive or negative, respectively.

Local processes include damping, equilibration in chemical reactions, dissipative effects, or coupling between equations. Generalizations of equation (4–2.1) also represent coupling between different evolution equations. Examples of coupled local processes are the drag terms in multiphase momentum equations. If ρ is a two-component vector consisting of an electron and an ion temperature, the generalization of equation (4–2.1) expresses thermal equilibration at the rate γ (here negative). If ρ is a vector of chemical reactants, equation (4–2.1) generalizes to the set of chemical kinetic rate equations and γ is a second-order tensor.

Now consider integrating the simplest case, the single ordinary differential equation, equation (4–2.1), with γ and S constant. This equation has an analytic solution that we

compare directly to numerical solutions derived from an explicit algorithm, an implicit algorithm, and an asymptotic algorithm, as defined below. The solution for each type of algorithm can be derived in closed form. We show that the most accurate and cost-effective technique within this framework is a hybrid algorithm consisting of an explicit (or centered) method and an asymptotic method. The explicit method is used when the stable timestep restriction is not prohibitive (normal equations). The asymptotic method is used when the timestep required for stability of the explicit algorithm is too small for practical applications (stiff equations).

When γ and S are constant, equation (4–2.1) has the analytic solution

$$\rho(t) = \left(\rho(0) + \frac{S}{\gamma}\right)e^{\gamma t} - \frac{S}{\gamma}, \tag{4–2.4}$$

whether γ is positive or negative. When γ is positive, the solution eventually grows exponentially from its initial value $\rho(0)$, though it may initially decrease and change sign. When γ is negative, the solution relaxes exponentially toward the asymptotic value $-S/\gamma$.

Equation (4–2.1) is really an ordinary differential equation, though we have been writing it with a partial derivative notation to remind us of the larger multidimensional reactive flow context. We can approximate equation (4–2.1) by a simple finite-difference formula

$$\frac{\rho^n - \rho^{n-1}}{\Delta t} = S + \gamma[\theta\rho^n + (1-\theta)\rho^{n-1}], \tag{4–2.5}$$

where

$$\rho^n \equiv \rho(n\,\Delta t), \tag{4–2.6}$$

and Δt is the fixed finite-difference timestep. The numerical values of ρ are produced only at discrete times $t^n = n\,\Delta t$. The implicitness parameter θ ranges from 0 to 1 and determines whether the right hand side of equation (4–2.5) is evaluated using $\rho(n\,\Delta t)$ at the new time ($\theta = 1$ is the fully implicit solution), at the old time using $\rho((n-1)\,\Delta t)$ ($\theta = 0$ is the fully explicit solution), or somewhere in between (semi-implicit solution).

The finite-difference equation, equation (4–2.5), can be solved for ρ^n in terms of the initial density $\rho(0)$. This formal solution,

$$\rho^n = \left(\rho(0) + \frac{S}{\gamma}\right)E(\gamma\,\Delta t) - \frac{S}{\gamma}, \tag{4–2.7}$$

is similar to equation (4–2.4) where $E(\gamma\,\Delta t)$ is the algorithm-dependent finite-difference approximation to the exponential function $e^{\gamma\Delta t}$. For the particular approximation equation (4–2.5), we find

$$E(\gamma\,\Delta t) \equiv \frac{1 + (1-\theta)\gamma\,\Delta t}{1 - \theta\gamma\,\Delta t}. \tag{4–2.8}$$

The highest-order approximation of the exponential $e^{\gamma t}$ occurs when $\theta = \frac{1}{2}$, but this is *not* the most accurate approximation. For any value of γ, a value of θ can be chosen which makes the approximation exact. This particular value, θ^*, is called an integrating factor;

here,

$$\theta^* = \frac{e^{\gamma\,\Delta t} - \gamma\,\Delta t - 1}{\gamma\,\Delta t(e^{\gamma\,\Delta t} - 1)}\,. \tag{4–2.9}$$

When $\theta = \frac{1}{2}$, equation (4–2.8) can be expanded as

$$E(\gamma\,\Delta t) \cong 1 + \gamma\,\Delta t + \frac{(\gamma\,\Delta t)^2}{2} + \frac{(\gamma\,\Delta t)^3}{4} + \cdots. \tag{4–2.10}$$

Here E is an accurate representation of the exponential to second order in $\gamma\,\Delta t$, with an error term of $-(\gamma\,\Delta t)^3/12$. When $\gamma\,\Delta t$ approaches zero, θ^* approaches $\frac{1}{2}$.

Figure 4.3 shows these solutions for the decaying case where $\gamma < 0$ and $S = 0$. The solutions should approach zero asymptotically starting with $\rho(0)$. These solutions represent a stable relaxation, which is characteristic of most coupling and chemical kinetic processes. The three panels show the results of numerical integrations using timesteps $-\gamma/2$, $-\gamma$, and $-2/\gamma$, where the characteristic relaxation time is $|\gamma|^{-1}$. The four solutions shown for each choice of timestep are labeled by the letters E for the explicit solution ($\theta = 0$), C for the centered solution ($\theta = \frac{1}{2}$), I for the fully implicit solution ($\theta = 1$), and T for the theoretical (analytical) solution.

When $-\gamma\,\Delta t$ is less than unity, as in Figure 4.3a, the explicit solution never changes sign. All of the numerical solutions replicate the qualitative behavior of the decaying exponential and are numerically stable. The centered solution (C) is nearly indistinguishable from the exact theoretical solution. The explicit solution (E) is lower than the exact solution (T) by about the same amount as the implicit solution (I) is too large. Unfortunately, ensuring $|\gamma|\,\Delta t \leq 1$ at each timestep can be expensive computationally when the equations are *stiff*. Here being stiff means that

$$\frac{-\gamma\rho}{\partial\rho/\partial t} \gg 1. \tag{4–2.11}$$

When $\gamma\,\Delta t = -1$ in Figure 4.3b, the centered solution still approximates the exact solution very well, although the implicit solution is noticeably less accurate than the solution obtained with a smaller timestep. The explicit solution displays a rather singular behavior: it goes exactly to zero in one timestep and then stays there. The asymptotic state, $\rho = 0$, has been achieved far too early. All solutions are stable and behave in a physically reasonable way.

In Figure 4.3c, the timestep is increased further so that $\gamma\,\Delta t = -2$. The explicit solution behaves strangely, oscillating at fixed amplitude about the asymptotic $\rho = 0$ state. This situation marks the borderline of mathematical instability for the explicit solution. As shown in panel (b), the explicit solution is incorrect even when Δt is a factor of two smaller, that is, $\Delta t = -1/\gamma$. For Δt larger than $-1/\gamma$, the value of ρ computed explicitly goes below zero and changes sign each timestep. This occurs because the approximation to $e^{\gamma t}$, $E(\gamma\,\Delta t)$ in equation (4–2.8), becomes negative.

Solutions of coupled systems of equations for real problems generally cannot tolerate the mathematically stable but physically bizarre behavior shown in Figure 4.3c. Chemical kinetics calculations, for example, are explosively unstable when species number densities have negative values. An important rule of thumb in computational physics is that the

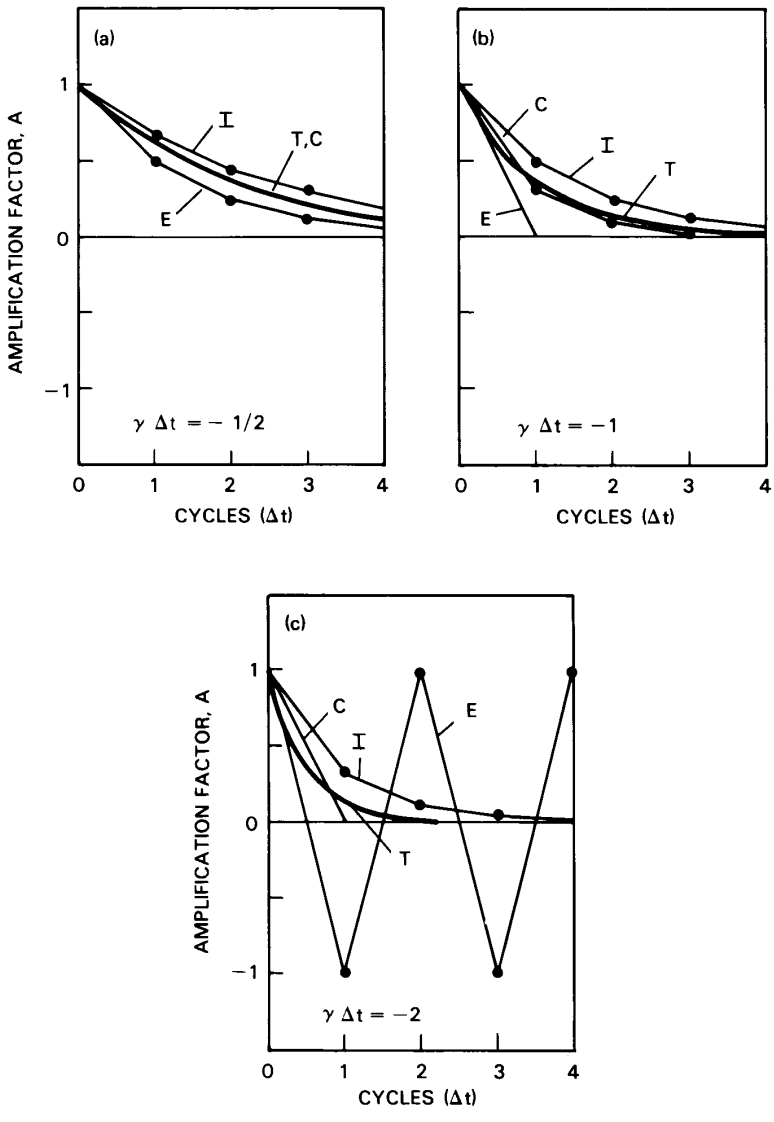

Figure 4.3. Solutions for a local scalar process for various values of the parameter $\gamma\,\Delta t$. The labels T, E, C, and I indicate the analytical (true), explicit, centered, and implicit solutions, respectively. (a) $\gamma\,\Delta t = -1/2$. (b) $\gamma\,\Delta t = -1$. (c) $\gamma\,\Delta t = -2$.

numerical timestep should always be chosen at least a factor of two smaller than required by the most stringent mathematical stability condition. In other words, practical stability, which gives the qualitatively correct physical behavior, is at least a factor of two more expensive in terms of computer requirements than mathematical stability. Not keeping the timestep well below the stability estimate is a frequent source of error in computations.

When $\gamma\,\Delta t = -2$, the centered solution still behaves somewhat acceptably, but reaches the limiting case $\rho = 0$ before instability occurs in the form of negative values of ρ. The implicit solution is behaving physically and stably, although it is not decaying quickly enough. Guaranteed stability is the lure of the implicit approach; unfortunately, the error

in the implicit solution, relative to the rapidly decaying theoretical curve, has grown even larger than in the previous two panels. This potentially large error in the implicit relaxation rate of stiff equations is the hidden sting. The practical timestep stability condition for an explicit method is also the effective accuracy condition for the implicit method.

4–2.2. Asymptotic Solutions

There are many situations where the decaying nature of an exponential relaxation makes large relative errors progressively less important because the correct equilibrium is being approached. There are also physically unstable situations where these same errors are more important. Although the implicit algorithm guarantees stability, it becomes considerably more expensive in complex systems of many coupled equations because the division sign in equation (4–2.8) becomes a matrix-inversion operation. Thus there are continuing efforts to develop less expensive methods to treat the case $|\mathcal{Y}|\,\Delta t \gg 1$. In an asymptotic expansion, mathematical accuracy is obtained when $|\mathcal{Y}|\,\Delta t$ is large. Thus it is distinguished clearly from the methods just described, which are based on a Taylor-series expansion and which converge when $|\mathcal{Y}|\,\Delta t$ is small.

Let both $\mathcal{Y}$ and S be time dependent. Variations in these quantities introduce nonlinear coupling and multiple time scales into the problem when several variables are involved. These effects are not major in the example discussed here, but they are important in the more detailed discussion in Chapter 5. We also assume that ρ relaxes rapidly towards its limiting value, but that the rate of approach keeps changing as $\mathcal{Y}(t)$ and $S(t)$ evolve.

To derive an asymptotic approximation to equation (4–2.1), the right hand side is set to zero and the derivative $\partial\rho/\partial t$ is iteratively calculated using the previous approximation. For example, consider equation (4–2.1) at time n,

$$\mathcal{Y}^n\rho^n = -S^n + \left.\frac{\partial\rho}{\partial t}\right|^n. \tag{4–2.12}$$

The first-order asymptotic solution is

$$^{(1)}\rho^n \approx -\frac{S^n}{\mathcal{Y}^n}, \tag{4–2.13}$$

assuming $\partial\rho/\partial t$ is negligible compared to the other terms. Including the time derivative of ρ in equation (4–2.13) gives the next approximation. This derivative can be approximated using the lowest-order solution just obtained,

$$\begin{aligned}\left.\frac{\partial\rho}{\partial t}\right|^n &\approx -\frac{1}{\Delta t}\left(\frac{S^n}{\mathcal{Y}^n} - \frac{S^{n-1}}{\mathcal{Y}^{n-1}}\right)\\ &\approx -\frac{1}{\Delta t}\left(\frac{S^n}{\mathcal{Y}^n} - \rho^{n-1}\right).\end{aligned} \tag{4–2.14}$$

Thus the next order of ρ can be written

$$^{(2)}\rho^n \approx -\frac{S^n}{\mathcal{Y}^n} - \frac{\left(\frac{S^n}{\mathcal{Y}^n} - \frac{S^{n-1}}{\mathcal{Y}^{n-1}}\right)}{\mathcal{Y}^n\,\Delta t}. \tag{4–2.15}$$

Note in equation (4–2.12) that the derivative $\partial\rho/\partial t$ is strictly evaluated at time n, but is taken at time $n-\frac{1}{2}$ in equation (4–2.14). This phase lag in the time at which derivatives can be evaluated is a characteristic of asymptotic numerical approximations. It often means that the lower-order approximations are actually more accurate than the higher-order asymptotic approximation.

An expansion such as equation (4–2.15) is adequate only when $|\gamma|\,\Delta t$ is large. For example, when $|\gamma|\,\Delta t = 2$,

$$^{(2)}\rho^n \approx -\frac{3}{2}\frac{S^n}{\gamma^n} + \frac{1}{2}\frac{S^{n-1}}{\gamma^{n-1}}. \tag{4–2.16}$$

When $|\gamma|\,\Delta t = 1$,

$$^{(2)}\rho^n \approx -2\frac{S^n}{\gamma^n} + \frac{S^{n-1}}{\gamma^{n-1}}. \tag{4–2.17}$$

When $|\gamma|\,\Delta t = \frac{1}{2}$,

$$^{(2)}\rho^n \approx -3\frac{S^n}{\gamma^n} + 2\frac{S^{n-1}}{\gamma^{n-1}}. \tag{4–2.18}$$

As $|\gamma|\,\Delta t$ becomes small, the value of the kth approximation, $^{(k)}\rho^n$, is dominated by an improperly computed time derivative.

Major gains in solution speed and accuracy are achieved by using a hybrid method. In particular, consider combining an asymptotic method when the equations are stiff and extremely short timesteps would be otherwise required with a second method to get high accuracy when the equations are not stiff. In the simple problem illustrated in Figure 4.3, the asymptotic formula gives the final relaxed state $\rho = 0$. Since the explicit and centered formulas are unstable for $\Delta t > 1/|\gamma|$ and $\Delta t > 2/|\gamma|$, respectively, the asymptotic formula should be used whenever Δt exceeds these values.

The maximum error made by using either the explicit or centered formulas for normal equations and the asymptotic formula for the stiff equations occurs at an intermediate value of Δt. With a combination of explicit and lowest-order asymptotic methods, the maximum error is e^{-1}. A centered-asymptotic combination has an error of e^{-2} at the transition value $\Delta t = 2/|\gamma|$.

Because the implicit formula is always stable, the transition point can be taken at any value of Δt. It is useful to determine where the two formulas, implicit and asymptotic, are most accurate. From equation (4–2.8), the error made in using the implicit formula for one timestep (taking $\theta = 1$) is

$$\text{Implicit error} = \text{implicit solution} - \text{exact solution} = \frac{1}{1-\gamma\,\Delta t} - e^{\gamma\Delta t}. \tag{4–2.19}$$

The corresponding error using the asymptotic formula is

$$\text{Asymptotic error} = -e^{\gamma\,\Delta t}, \tag{4–2.20}$$

because the lowest order asymptotic solution is zero. The magnitude of these errors are equal when $-\gamma\,\Delta t \approx 1.679$. For timesteps less than $\Delta t \approx -1.679/\gamma$, the implicit formula

is more accurate. For longer timesteps, the asymptotic formula is more accurate. The implicit-asymptotic combination has a worst error of $e^{-1.679}$. Because the worst error of the centered-asymptotic combination is smaller than that of the implicit solution and the computational time is the same, there really is no reason to use a fully implicit technique.

The material presented above introduced the concepts of implicit, explicit, centered, and asymptotic methods. The discussion of these concepts will be expanded in Chapter 5, which discusses methods for solving coupled nonlinear ordinary differential equations. Asymptotic methods are good for stiff regions of the solution but not as good for following rapid variations. They are most useful when coupled to fluid dynamics equations. Asymptotic methods are also useful for treating sound waves when they are considered as stiff phenomena, a topic that is discussed in conjunction with the slow-flow algorithm in Chapter 9.

4–3. Diffusive Transport

4–3.1. The Diffusion Equation

Diffusion, the second process in Table 4.1, is a nonlocal process with spatial as well as temporal variations. The general multidimensional diffusion equation

$$\frac{\partial}{\partial t}\boldsymbol{\rho}(\mathbf{x}, t) = \nabla \cdot \mathbf{D}(\mathbf{x}, t, \boldsymbol{\rho})\nabla\boldsymbol{\rho}(\mathbf{x}, t), \tag{4–3.1}$$

where $\boldsymbol{\rho}$ and $\mathbf{x}$ are vectors, is simplified here to a single, one-dimensional, constant-coefficient diffusion equation,

$$\frac{\partial}{\partial t}\rho(x, t) = D\frac{\partial^2}{\partial x^2}\rho(x, t). \tag{4–3.2}$$

Here D is assumed constant in space and time and thus can be taken out of the divergence operator. By analogy with equation (4–1.1), a spatially and temporally varying source term in the form of $S(x, t)$ can be added to the right-hand side of equations (4–3.1) and (4–3.2). This extra term is zero here, so the steady-state solution of equation (4–3.2) has the constant value ρ^∞, determined by conservation throughout the computational domain.

The diffusive flux, defined as

$$F_x \equiv -D\frac{\partial\rho}{\partial x}, \tag{4–3.3}$$

is proportional to the gradient of the fluid variable $\rho(x, t)$. The sign of the flux is such that large values of ρ tend to decrease as regions with lower values of ρ fill in.

There are two cases where equation (4–3.3) is not a good representation of the diffusion process. In the first case, the breakdown is at very small scales, where variations are appreciable over only a few mean free paths. These large variations occur when we try to represent the random walk of atoms and molecules as a continuous diffusion term on a macroscopic scale. The equation also breaks down at very large scales, which is a numerical effect of the form of the term. If the diffusion flux of ρ, equation (4–3.3), is divided by the value of ρ at a location x far from a source, the characteristic diffusive transport velocity of ρ is

$$v_c = \frac{x}{2t}. \tag{4–3.4}$$

This says that for any finite time, the diffusive transport velocity becomes arbitrarily large at distances far enough away from the initial source of ρ. These major anomalies are considered further in Chapter 7.

4–3.2. An Analytic Solution: Decay of a Periodic Function

To assess numerical solution methods, we use an idealized problem for which there is an analytic solution to equation (4–3.2). As with the simplified ordinary differential equation methods discussed in Section 4–2, this idealized problem allows us to compare the "exact" results with those from approximate finite-difference algorithms. It also permits amplitude errors for various algorithms to be compared analytically.

Consider the decay in time of periodic, sinusoidal profiles $\rho(x, t)$, which initially have the amplitude $\rho(x, 0)$. The spatial structure is fixed in shape because sines and cosines are eigenfunctions of equation (4–3.2), but the amplitude changes in time. We use the complex exponential trial function

$$\rho(x, t) = \rho(t)(\cos kx + i \sin kx) \tag{4–3.5}$$

in equation (4–3.2). The part of ρ containing the time variation, designated here by $\rho(t)$, can be found assuming that the wavenumber k is constant.

The complete analytic solution to equation (4–3.2),

$$\rho(x, t) = \rho_0 e^{-Dk^2 t} e^{ikx}, \tag{4–3.6}$$

shows exponential decay in time of the sine and cosine spatial variations. Because the time-varying coefficient $e^{-Dk^2 t}$ is real, the phase of each harmonic is not affected by the diffusion operator. Sine and cosine components stay separate. Each Fourier harmonic of the profile decays in such a way that the total amount of ρ is conserved. The conservation integral is

$$\frac{d}{dt}\left\{\int_0^{2\pi/k} \rho(x, t)\, dx\right\} = 0, \tag{4–3.7}$$

where the function $\rho(x, t)$ is integrated over one wavelength. For equation (4–3.6), this conservation integral constraint also means that a solution, which initially has a real value of $\rho(0)$ everywhere, stays real. The zero phase shift property implies a symmetry requirement on the algorithm chosen to represent the spatial derivative. Because the time dependence is a real exponential, the sign of the decaying wave cannot change and positivity must be preserved in a reasonable solution.

Any numerical algorithm for the diffusion equation should reproduce the following fundamental qualitative properties:

1. The total integral of $\rho(x, t)$ should be conserved.
2. The amplitude, $|\rho(t)|$, should decay monotonically.
3. There should be no phase errors introduced by the algorithm (such as might arise from asymmetric forms of the derivative).
4. Positivity should be preserved.

4–3.3. Explicit and Implicit Solutions

One way to write equation (4–3.2) in simple finite-difference form is

$$\frac{\rho_j^n - \rho_j^{n-1}}{\Delta t} = \frac{D\theta}{(\Delta x)^2}\left[\rho_{j+1}^n - 2\rho_j^n + \rho_{j-1}^n\right] + \frac{D(1-\theta)}{(\Delta x)^2}\left[\rho_{j+1}^{n-1} - 2\rho_j^{n-1} + \rho_{j-1}^{n-1}\right]. \tag{4–3.8}$$

This formula is similar to equation (4–2.5) used to approximate local processes, and the time derivative is differenced identically. The space derivative is evaluated using a fraction θ of the values of $\{\rho\}$ at the new time t^n and a fraction $(1-\theta)$ of the values at the beginning of the timestep, t^{n-1}. Again, we select three specific values of θ in the range $0 \le \theta \le 1$. When $\theta = 0$, the method is explicit, denoted E, and new values of ρ depend only on previous values of ρ. The explicit algorithm is computationally the least expensive to use because the new values are not coupled to each other. When $\theta = \frac{1}{2}$, the resulting algorithm is called centered, denoted by C. The centered algorithm has a higher mathematical order of accuracy than algorithms for which θ is not equal to $\frac{1}{2}$. The third choice, $\theta = 1$, is the most stable, fully implicit algorithm denoted by I.

As in the theoretical example given in equations (4–3.5) and (4–3.6), we assume that ρ initially has a sinusoidal spatial variation which can be written as e^{ikx}. For each separate mode, equation (4–3.8) yields

$$\begin{aligned}\frac{\rho^n(k) - \rho^{n-1}(k)}{\Delta t} &= -\frac{2D\theta}{(\Delta x)^2}(1 - \cos k\,\Delta x)\rho^n(k) \\ &\quad - \frac{2D(1-\theta)}{(\Delta x)^2}(1 - \cos k\,\Delta x)\rho^{n-1}(k).\end{aligned} \tag{4–3.9}$$

Values of $\rho(k)$ at both the new timestep n and old timestep $n-1$ appear on both sides of the equation. Equation (4–3.9) can be solved for $\rho^n(k)$ in terms of the value at the previous timestep $\rho^{n-1}(k)$ because we consider only one harmonic k at a time. In the general case, equation (4–3.8) can be solved by a tridiagonal matrix inversion algorithm.

In terms of A, the single-timestep amplification factor defined in equation (4–1.12), the finite-difference solution to equation (4–3.9) can be written as

$$A^{FD}(k) = \frac{\left[1 - \frac{2D\Delta t}{(\Delta x)^2}(1-\theta)(1 - \cos k\,\Delta x)\right]}{\left[1 + \frac{2D\Delta t}{(\Delta x)^2}\theta(1 - \cos k\,\Delta x)\right]}. \tag{4–3.10}$$

Here the superscript FD indicates the finite-difference solution. Over timestep Δt, the analytic solution given by equation (4–3.6) predicts the amplification factor

$$A^T(k) = \left(\frac{\rho^n}{\rho^{n-1}}\right)^T = e^{-Dk^2\Delta t}. \tag{4–3.11}$$

The superscript T designates the theoretical or true solution to the sine wave test problem posed above. The amplitude of each mode should decay exponentially at the rate Dk^2.

We now want to determine the range of wavenumbers k and timesteps Δt over which the individual numerical algorithms work well. To do this, compare the solution obtained

from the explicit, centered, and implicit numerical algorithms ($\theta = 0$ (E), $\theta = \frac{1}{2}$ (C), and $\theta = 1$ (I)) with the theoretical amplification factor in equation (4–3.11). Two nondimensional parameters are useful,

$$\beta \equiv \frac{2D\Delta t}{(\Delta x)^2} \quad \text{and} \quad k\,\Delta x.$$

Here β is a measure of the size of the timestep relative to how fast gradients diffuse on the scale of a cell size Δx. The other parameter, $k\,\Delta x$, is the wavenumber of the harmonic being considered in units of the cell size. The values of $k\,\Delta x$ range from 0 to π (two cells per wavelength), and the values of β generally range from zero to infinity.

The amplification factors for the three finite-difference algorithms, given by equation (4–3.10), and the theoretical exact solution, equation (4–3.11), may be rewritten as

$$A^{FD}(k) = \frac{[1 - \beta(1-\theta)(1-\cos k\,\Delta x)]^{1/\beta}}{[1 + \beta\theta(1-\cos k\,\Delta x)]^{1/\beta}} \tag{4–3.12}$$

and

$$A^{T}(k) = e^{-\frac{1}{2}(k\,\Delta x)^2}, \tag{4–3.13}$$

respectively. The theoretical result in equation (4–3.13) is the $(1/\beta)$th root of equation (4–3.6) and expresses the amplification factor per unit time rather than per timestep. The exponential factor $1/\beta$ is included in the definition of equation (4–3.12) so that comparisons of various solution algorithms may be made at fixed times even though different timesteps are used. The exact theoretical solution is compared with the approximate solutions in Figure 4.4.

Figure 4.4 shows A as a function of $k\,\Delta x$ for various values of β. The far right of each panel is the shortest wavelength that can be represented, two cells per wavelength. The far left is the infinite wavelength limit. The points where the explicit solution reflects from the horizontal axis where $A = 0$ indicate the part of the short-wavelength spectrum where the mode amplitude changes sign because the timestep is too long. This occurs first for solution E when $\beta = 1$ and subsequently for solution C at $\beta = 1.5$. For $k\,\Delta x$ larger than the reflection value, the corresponding algorithms predict negative values of the positive quantity ρ. Thus these reflection points are the practical, if not mathematical, limit of numerical stability.

For Figure 4.4c, where $\beta = 1.5$, the short-wavelength portion of the spectrum computed using the explicit algorithm is unstable: the amplitude is greater than unity and the sign is wrong. For $k\,\Delta x \geq 2\pi/3$, the centered algorithm is practically, although not absolutely, unstable. The amplitudes of these short-wavelength components of an overall solution decrease rapidly from one timestep to the next. This is the correct trend, but their sign oscillates unphysically.

The best method for solving the diffusion equation depends on the diffusion coefficient and the affordable timestep for a given cell size. Equation (4–3.12) should be equal to the correct answer, equation (4–3.13), when θ is properly chosen. It is not possible to find one integrating factor as we could for equation (4–2.5) because each harmonic of the

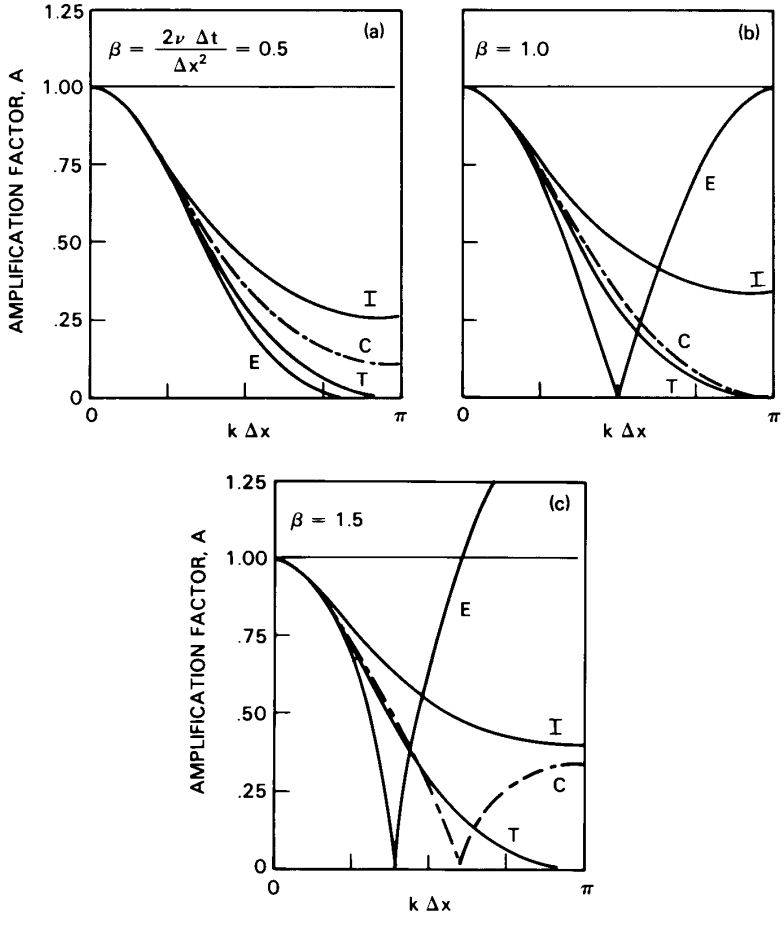

Figure 4.4. The amplification factor, A, for a diffusion equation, as a function of $k\ \Delta x$ for various values of the parameter β. The T, E, C, and I refer to the theoretical (analytical), explicit, centered, and implicit solutions, respectively. (a) $\beta = 0.5$. (b) $\beta = 1.0$. (c) $\beta = 1.5$.

spatial variation requires a different value $\theta^*(\Delta t)$. In the long-wavelength limit, however, the spatial mode number variation of θ^* drops out and

$$\theta^* \approx \frac{3\beta - 1}{6\beta} \,. \tag{4–3.14}$$

When the nondimensional timestep β is small, θ^* in equation (4–3.14) is negative, which means that the explicit solution with $\theta = 0$ is most accurate for the values $\beta \leq \frac{1}{3}$. This is a regime in which no sensible algorithm is exact because the correct answer is not attainable with a value of θ in the range 0 to 1. Thus there is *no* regime where the fully implicit algorithm is the most accurate algorithm. Further, when the explicit algorithm is stable for all wavelengths, it is also the most accurate algorithm at long wavelengths.

The explicit algorithm is the fastest and most accurate whenever a short enough timestep can be selected. When longer timesteps – and thus larger values of β – are required, a

partially implicit algorithm with θ chosen according to equation (4–3.14) should be used. For wavelengths shorter than four cells, strong additional damping must be added to ensure the positivity of the composite algorithm.

4–3.4. The Cost of Implicit Methods

In all of these comparisons, the implicit algorithm is stable and behaves in a physically reasonable way at short wavelengths even though it is never the most accurate algorithm. Strict conservation of ρ is satisfied because equation (4–3.8) can be written in flux conservation form. In the regimes where the other algorithms are also stable, the implicit diffusion algorithm is somewhat less accurate but this should not be a major concern. Short-wavelength parts of the solution, where numerical errors are the worst, decay quickly and the solution becomes smoother as time progresses. All of these considerations indicate that implicit diffusion algorithms may be used wherever practical.

There are, however, other drawbacks to using partially or fully implicit algorithms. For example, there are numerical complications when we have to deal with more than one spatial dimension, with nonlinearity from coupled diffusion equations, and with spatial dependence of the transport coefficients. Solving the multidimensional diffusion equation implicitly involves inverting large, sparse matrix equations of the type discussed in Section 10–4. When the equations are linear, this is straightforward although computationally expensive. When the equations are nonlinear, the functional form of the coefficients is complicated and iteration of linearized matrix solutions is required. Then the expense can be very high. It is possible to use an asymptotic solution in the limit of very fast diffusion, and this also involves solving a large matrix. An asymptotic solution should again be more accurate than the implicit formulation, as was the case for local processes discussed above, but may not maintain conservation.

4–3.5. Physical Diffusion and Numerical Diffusion

Numerical diffusion, a major source of error in finite-difference algorithms, has the same form and effect as physical diffusion. Figure 4.2a is a schematic diagram showing how a peaked profile $\rho(x, t)$ spreads because of numerical diffusion. The three curves in panel (a) describe equally well the analytic solution of the evolution of a point-source profile at three successive times, as determined by equation (4–3.7). As the profile decreases in amplitude, it becomes broader. For numerical diffusion, as for physical diffusion, the amplitudes of the solution harmonics change but the phases do not.

To complete the analogy between numerical diffusion and equations (4–3.2) and (4–3.7), suppose that the direction of time is reversed in these equations. Graphically this is shown in Figure 4.2a with $t_1 > t_2 > t_3$. The solution evolves the way many strong numerical instabilities appear to behave in simulations. When these algorithms go unstable, their behavior can be analyzed to show that there has been a change of sign in the diffusion term. Taking too big a timestep is a simple way to induce this. A Gaussian profile run backward in time by an "antidiffusion" equation shows increasing spikiness. As the profile gets narrower, its height increases until it becomes singular at some finite time. Diffusion equations, when reversed in time, are both mathematically and physically

unstable. The δ-function singularity which we have shown at $t = 0$ is a typical result of this property.

In Chapter 7, we return to the δ-function diffusion problem and discuss it from a more physical perspective. Here, however, we note some important numerical implications of the tendency of a localized diffusing system to approach a Gaussian. If errors in the solution tend to decrease in time as the calculation proceeds, the solution becomes more accurate rather than less accurate. The short-wavelength components of the solution diffuse the fastest, and their amplitudes quickly approach zero if there are no sources of short-wavelength structure. This is convenient because numerical simulations using finite spatial grids usually have the worst errors at short wavelengths. If the physical diffusion is large enough, it actually reduces the short-wavelength errors generated by numerical representations with finite resolution.

Because real diffusion smooths profiles, an evolving solution loses track of details in initial conditions and any fluctuations subsequently added by truncation or round-off errors. In a finite time, short-wavelength components in the numerical solution decay away. Thus solution algorithms with numerical diffusion can actually become more accurate, in an absolute sense, as the integration proceeds.

The phrases *diffusive effects* and *dissipative effects* are often used interchangeably. Diffusion and dissipation, in the thermodynamic sense, both imply entropy increases. In addition, "dissipative" has connotations which do not apply to diffusion. Characterizing a system as dissipative implies that the system is irreversible and has losses. This type of process is usually not conservative. Diffusion, however, while thermodynamically irreversible (that is, entropy goes up during the diffusive mixing of two media), is fully conservative in that the total amount of ρ stays the same even though its distribution in space changes.

4–4. Convective Transport

4–4.1. Fluid Dynamic Flows: Advection and Compression

The continuity equation, the third process in Table 4.1, describes the compression, rotation, and translation (advection) of a fluid. These three effects together constitute convection, that is, continuous fluid dynamic flow. In the ideal one-dimensional limit, the equation

$$\frac{\partial}{\partial t}\rho(x,t) = -\frac{\partial}{\partial x}\rho(x,t)v(x,t) \tag{4–4.1}$$

describes the convection of a quantity $\rho(x,t)$ by a flow velocity $v(x,t)$. Limiting the problem to one dimension excludes rotations, but allows both advection and compression. (See the discussion in Section 2–2.1.)

Solving continuity equations accurately is an important part of computational fluid dynamics. Fluid dynamics is governed by coupled continuity equations: one scalar continuity equation for the mass density ρ; one vector continuity equation with one, two, or three components for the momentum density ρv; and one scalar continuity equation for the energy density E. The discussion in this section, of a single, one-dimensional continuity equation, is a starting point for the treatment of more sophisticated solution techniques for coupled and multidimensional continuity equations.

The one-dimensional, Cartesian continuity equation is written in *conservation form* in equation (4–4.1). It can also be written in *Lagrangian* form,

$$\frac{\partial \rho}{\partial t} + v\frac{\partial \rho}{\partial x} \equiv \frac{d\rho}{dt} = -\rho\frac{\partial v}{\partial x}. \tag{4–4.2}$$

The two terms on the left hand side of equation (4–4.2) are the *Eulerian time derivative* and the *advective space derivative*, respectively. Together they comprise the *Lagrangian derivative*, $d\rho/dt$, which gives the rate of change of the density variable ρ in a frame of reference that moves with the local fluid velocity. This representation is useful because the Lagrangian derivative is only nonzero because of the compression term, $v\partial\rho/\partial x$. Advection and rotation do not appear explicitly in the Lagrangian representation.

The compression term also includes the inverse effect, rarefaction. Compression contains a spatial derivative of the velocity, $\partial v/\partial x$ in one dimension and the full divergence operator in multidimensions, but it is linear in the convected quantity ρ. The net effect of compression is local because the gradient of ρ does not appear. If we treat $\partial v/\partial x$ as a known quantity while solving equation (4–4.2), compression is just one of the local effects discussed in Section 4–2. To simplify the discussion of advection in this section, we eliminate compressional effects by assuming the velocity is constant, so that $\partial v/\partial x = 0$. This allows use of Fourier analysis to study advection where the short-wavelength components of the phase error become extremely important.

Equation (4–4.2) suggests numerical algorithms for the continuity equation which take advantage of a Lagrangian frame of reference that moves with the fluid. In a Lagrangian representation, fluid parcels (cells) in the discrete model have time-varying boundaries that move with the local fluid velocity. These cell boundaries are defined so that the convective flux into or out of the cell is zero. Thus the mass of a Lagrangian fluid parcel is constant. In the Lagrangian frame of reference, the troublesome convective derivatives, the source of numerical diffusion, are no longer in the equations.

Lagrangian approaches seem to be an excellent way to eliminate major advection errors, and this is true for some one-dimensional problems. Unfortunately, multidimensional Lagrangian fluid algorithms are more difficult than Eulerian algorithms because the moving grid becomes tangled as the flow evolves. Points that were originally close in a complicated flow can move quite far from each other. To carry Lagrangian simulations more than a few timesteps, the grid must be restructured or reconnected. Thus the geometry of the distorted Lagrangian grid enters into the problem and makes the calculations difficult and expensive. Another complication is that it is not possible to treat all of the compressible reactive flow variables in the same Lagrangian frame of reference, even in one dimension. Because of the complications associated with Lagrangian representations, this book concentrates on Eulerian methods that we find to be most robust, especially for multidimensional problems.

As with local processes and diffusion equations (Sections 4–2 and 4–3), the major errors in representing advective effects are caused by discretizing the continuous variables. In modeling convection, discretization introduces phase errors, amplitude errors, and Gibbs errors. Thus solving continuity equations combines the difficulties of the diffusion processes described above and the wave processes considered in Section 4–5. A perfect

method for convection would transport the quantity ρ at the correct velocity, maintain the correct amplitudes, and preserve the steep gradients at the correct locations.

4–4.2. The Square-Wave Test Problem

Figure 4.5 shows an application of the continuity equation that is very useful for testing numerical solution methods. A cloud of material is injected into a background gas and drifts along with it. The location and shape of the cloud are shown in Figure 4.5a at two different times, t_1 and t_2. In one dimension, the density of the cloud can be modeled by a square wave, as shown in Figures 4.5b and c, if

- the cloud is initially deposited over a finite region;
- the flow is incompressible, so that the density of the two materials is nearly constant; and
- there is essentially no diffusion present, so that the materials do not interpenetrate.

Figures 4.5b and c show one-dimensional density profiles of ρ, measured through the center of the cloud along the flow direction. The uniform velocity convects the cloud a

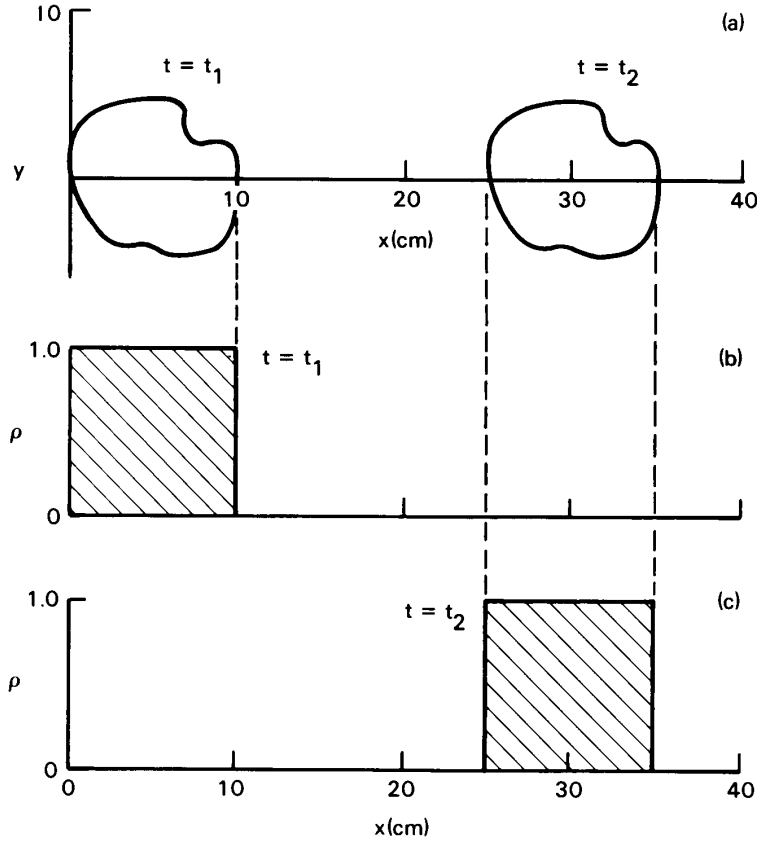

Figure 4.5. Schematic for the square-wave test problem. (a) A cloud of gas moves at a constant velocity, unaffected by diffusion. (b) Density along the centerline of the cloud at the initial time t_1. (c) Density along the centerline of the cloud at a later time t_2.

distance equal to about twice its spatial extent after $t_2 - t_1$ seconds have elapsed. The sharp edge of the profile represents the narrow region at the edge of a cloud where the mixture varies rapidly from nearly pure cloud material, to a thin region of equally mixed cloud and background, to a much larger region of nearly pure background fluid.

The square density profile from $x = 0$ to $x = 10\,\text{cm}$ at time $t = 0$ is a good test of algorithms designed to calculate convection. It is a stringent test because all wavelengths of the Fourier expansion are involved in the solution and the correct phase relationship among all of the harmonics is important for maintaining the correct shape of the profile. Because v is constant, the true profile $\rho(x)$ moves through space without any relative distortion and the ρ square wave is centered at $x_2 = vt_2$, as shown in Figure 4.5c.

Many finite-difference methods, of varying degrees of sophistication, have been explored to solve equation (4–4.2). These range from the explicit, first-order, "donor-cell" algorithms to second-order methods such as the Lax-Wendroff, "leapfrog," or TVD algorithms (see, for example, Roache [1982]; Anderson [1995]; and Tannehill, Anderson, and Pletcher [1997]), to the higher-order, nonlinear monotonic methods (such as FCT, MUSCL, PPM, etc.). These methods are described and discussed in Chapters 8 and 9.

4–4.3. Explicit and Implicit Solutions

When $\partial v/\partial x = 0$, equation (4–4.2) reduces to the one-dimensional advection equation

$$\frac{\partial}{\partial t}\rho(x,t) = -v\frac{\partial}{\partial x}\rho(x,t). \tag{4–4.3}$$

This equation is linear in ρ with first-order derivatives in both space and time. Despite the obvious fact that the spatial derivative is first rather than second order, this equation looks rather like a diffusion equation. Similar finite-difference approximations are used now to study the advection equation.

As above, let Δx be the cell size and Δt be the fixed timestep interval. Then ρ_j^n is defined here as the average value of ρ in cell j at time $t^n = n\Delta t$. It can be represented over the interval Δx at time Δt as

$$\rho_j^n \equiv \int_{x_j-\Delta x/2}^{x_j+\Delta x/2} \rho(x, n\,\Delta t)\frac{dx}{\Delta x}. \tag{4–4.4}$$

In this representation the conserved sum is

$$\sum_{j=1}^{J}\rho_j^n\,\Delta x = \sum_{j=1}^{J}\rho_j^{n-1}\,\Delta x + \int_{(n-1)\,\Delta t}^{n\,\Delta t} v\left[\rho\left(x_J + \frac{\Delta x}{2}\right) - \rho\left(x_1 - \frac{\Delta x}{2}\right)\right]dt, \tag{4–4.5}$$

which replaces the corresponding conservation integral. The last two terms on the right hand side of equation (4–4.5) are the fluxes of ρ through the boundaries of the computational domain, taken to be at $x_{\frac{1}{2}} \equiv \Delta x/2$ and $x_{J+\frac{1}{2}} \equiv (J + \frac{1}{2})\Delta x$.

We use centered time and space differences with the first-order derivative formula to represent the advection term. As for equations (4–2.5) and (4–3.8), a finite-difference

equation approximating equation (4–4.3) is

$$\frac{\rho_j^n - \rho_j^{n-1}}{\Delta t} = -v\left[\theta\frac{\left(\rho_{j+1}^n - \rho_{j-1}^n\right)}{2\,\Delta x} + (1-\theta)\frac{\left(\rho_{j+1}^{n-1} - \rho_{j-1}^{n-1}\right)}{2\,\Delta x}\right] \tag{4–4.6}$$

and θ is again the implicitness parameter. The explicit algorithm (E) has $\theta = 0$, the centered algorithm (C) has $\theta = \frac{1}{2}$, and the fully implicit algorithm (I) has $\theta = 1$. These choices are of interest because the explicit algorithm is again the easiest to implement and least expensive, the time-centered algorithm generally has the highest order of accuracy, and the implicit algorithm is the most stable numerically.

Equation (4–4.6) can be simplified by combining timestep, cell size, and flow velocity into a single nondimensional parameter,

$$\epsilon \equiv v\,\frac{\Delta t}{\Delta x}. \tag{4–4.7}$$

Here ϵ is the number of cells of size Δx that the fluid traverses in a timestep Δt. The value of ϵ should usually be less than unity because the finite-difference formula uses information only from adjacent cells. Using this definition of ϵ in equation (4–4.6) yields

$$\rho_j^n + \frac{\theta\epsilon}{2}\left(\rho_{j+1}^n - \rho_{j-1}^n\right) = \rho_j^{n-1} - \frac{(1-\theta)}{2}\,\epsilon\left(\rho_{j+1}^{n-1} - \rho_{j-1}^{n-1}\right). \tag{4–4.8}$$

This equation gives the values of ρ at the new timestep n (on the left hand side) in terms of the values at the previously calculated timestep $n-1$ (on the right hand side). Equation (4–4.8) contains values of ρ^n at three different locations for any $\theta > 0$, so that a tridiagonal matrix must be solved at each timestep. When $\theta = 0$, this reduces to a simple explicit algebraic expression for ρ^n in terms of ρ^{n-1}. When the tridiagonal matrix must be solved at each timestep, the cost can be substantially larger than that of the explicit algorithm.

Figure 4.6 summarizes the results obtained using several methods to solve equation (4–4.3) for the square-wave test problem introduced in Figure 4.5. The top panel of Figure 4.6 shows the exact solution of the test problem at four different times. Note that the edges of the square wave are not perfectly vertical. This reflects the fact that the variables $\{\rho_j\}$ are defined by only a discrete set of numbers.

The test problem has a uniformly spaced grid of 100 cells extending from $x = 0$ to $x = 100$ cm. Initially $\rho = 1$ in the first ten cells and is zero everywhere else. The boundary conditions are periodic, so material flowing out of cell 100 through the interface at $x = 100$ cm reenters the computational region at the same speed and density through the interface to cell 1 at $x = 0$ cm. The flow velocity is a constant, $v = 1$ cm/s. With the timestep fixed at $\Delta t = 0.2$ s, the fluid moves 2/10 of a cell per timestep, and $\epsilon = 0.2$. At $t = 0$, the computed and exact solutions are identical by definition. The computed solutions obtained from equation (4–4.8) for later times are indicated by data points in Figure 4.6a through c.

Figure 4.6a shows the solution calculated with the explicit algorithm, equation (4–4.8) with $\theta = 0$, after 100 timesteps. Unfortunately this simple, inexpensive algorithm gives terrible results. It shows large oscillations that get worse in time, as well as errors that have propagated back onto the far right side of the grid due to the periodic boundary conditions. The results show little resemblance to the exact solution and are numerically unstable.

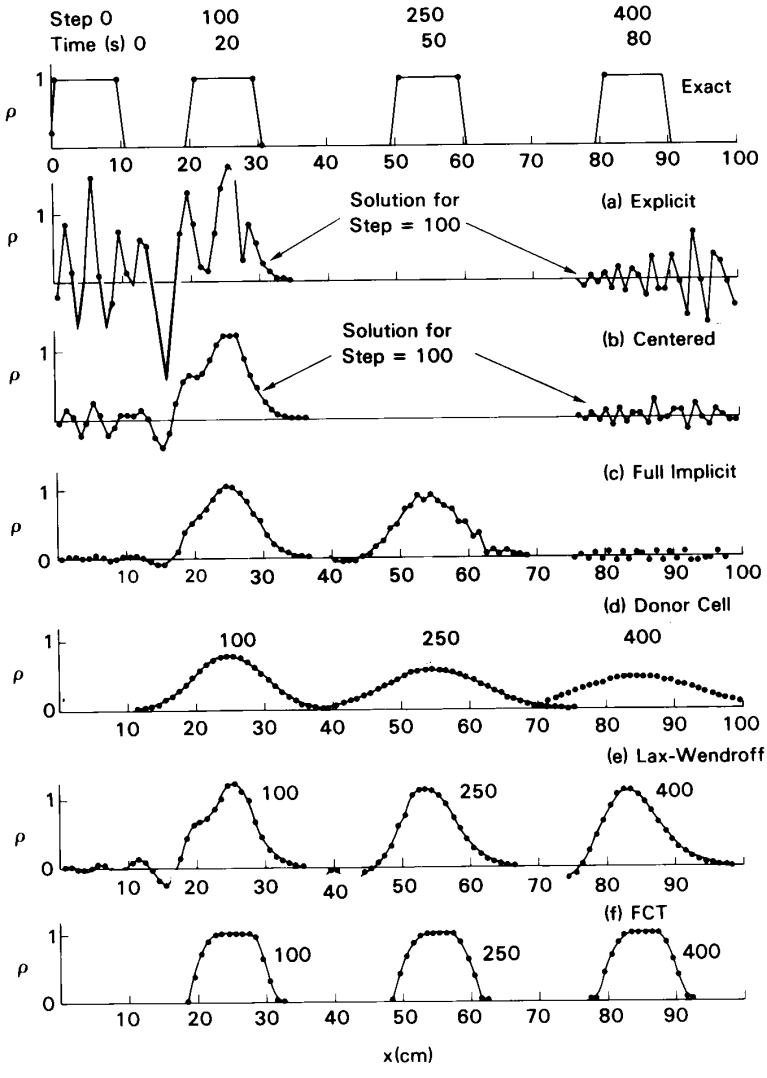

Figure 4.6. Solutions of the square-wave test problem using six different numerical algorithms. The top panel is the square wave as interpreted on the numerical grid described in the text. (a) Explicit algorithm, at step 100. (b) Centered algorithm, at step 100. (c) Fully implicit algorithm, at steps 100 and 250. Note that the noise on the far right corresponds to part of the solution at step 100. By step 250, the noise that has come in through the periodic boundary interacts with the main part of the solution. This gives the ragged appearance. (d) Donor cell, at steps 100, 250, 400. (e) Lax-Wendroff, at steps 100, 250, 400. (f) Flux-corrected transport (FCT), at steps 100, 250, 400 (discussed in Chapters 8 and 9).

No matter how small the timestep used, this kind of forward-differenced convection is unstable.

The centered solution is shown in Figure 4.6b. Because this algorithm is time reversible, it gives somewhat better results than the explicit algorithm. Mode amplitudes neither increase nor decrease in time. Though only a linear result, this is a highly desirable property. The centered solution also has higher-order accuracy than the explicit or fully implicit solutions. Nevertheless, the profile is not even roughly correct and there are large

nonphysical positive and negative values of density where ρ should be zero. The errors in the centered solution arise from a combination of Gibbs errors, which are bounded, and phase errors, which grow steadily in time.

Figure 4.6c was computed using the implicit algorithm. It has larger phase errors than the centered algorithm, but looks somewhat smoother. When the spatial difference is evaluated at the new time t^n, numerical damping is put into the solution. This damping appears to compensate partially for the larger phase errors. Neither the explicit, centered, nor fully implicit algorithms are really useful, however, as can be seen by the large difference between the exact and the computed solutions.

Figure 4.6d and e were computed using two other explicit, linear algorithms which are stable. Figure 4.6d is computed using the *one-sided* (also called *upwind* or *donor-cell*) algorithm which is only first-order accurate. Figure 4.6e shows the *Lax-Wendroff* algorithm, which is second-order accurate. Both of these algorithms are in common use, and both, as can be seen from the figure, have severe limitations.

For the donor-cell method, ρ at the new time t^n is calculated from

$$\rho_j^n = \rho_j^{n-1}(1-\epsilon) + \epsilon\, \rho_{j-1}^{n-1}. \tag{4–4.9}$$

Notice that the spatial derivative is approximated using two values of ρ, one of which is located upstream of the grid point j. This is the origin of the name "donor cell": the space derivative is computed using values associated with the fluid parcel which is being "donated" to cell j by the flow. When the velocity is negative and the flow is coming from the right, the donor-cell fluxes have to be calculated using the j and $j+1$ values of ρ. In this case the fluid arrives at cell center j at time t^n from the cell at $j+1$. Because the derivative is not centered and uses values from only two cells, the algorithm is accurate only to first order in space.

This linear interpolation in equation (4–4.9) introduces very severe numerical diffusion that overshadows the dispersive errors. The initially sharp edges of the density profile, which should propagate as sharp edges, smooth out rapidly. By step 500, more than half the material has numerically diffused out of the square wave. This occurs even though the square wave is resolved by 10 computational cells. In many practical three-dimensional calculations, a resolution of $50 \times 50 \times 50$ cells may be affordable on the computer available. Then the effect of numerical diffusion is relatively more severe than in the illustrative test case, Figure 4.6d.

Figure 4.6e shows the square-wave test performed using the Lax-Wendroff algorithm. One way to derive this algorithm is to use a second-order Taylor-series expansion in the time variable,

$$\rho(x,t^n) \cong \rho(x,t^{n-1}) + \Delta t \frac{\partial}{\partial t}\rho(x,t^{n-1}) + \frac{1}{2}(\Delta t)^2 \frac{\partial^2}{\partial t^2}\rho(x,t^{n-1}), \tag{4–4.10}$$

to estimate ρ at time $t^n = t^{n-1} + \Delta t$, given the values of $\{\rho_j\}$ at t^{n-1}. The next step is to recast the first and second time derivatives in terms of space derivatives, and then in terms of finite differences using equation (4–4.3). The derivatives in equation (4–4.10) become

$$\frac{\partial}{\partial t}\rho_j^{n-1} \cong -v\frac{\left(\rho_{j+1}^{n-1} - \rho_{j-1}^{n-1}\right)}{2\,\Delta x} \tag{4–4.11}$$

and

$$\frac{\partial^2}{\partial t^2}\rho_j = v^2\frac{\left(\rho_{j+1}^{n-1} - 2\rho_j^{n-1} + \rho_{j-1}^{n-1}\right)}{(\Delta x)^2}, \tag{4–4.12}$$

when v is a constant. Substituting equations (4–4.11) and (4–4.12) into equation (4–4.10), we obtain

$$\rho_j^n = \rho_j^{n-1} - \frac{\epsilon}{2}\left(\rho_{j+1}^{n-1} - \rho_{j-1}^{n-1}\right) + \frac{1}{2}\epsilon^2\left(\rho_{j+1}^{n-1} - 2\rho_j^{n-1} + \rho_{j-1}^{n-1}\right), \tag{4–4.13}$$

the simplest form of the Lax-Wendroff algorithm.

The results of the two first-order algorithms shown in Figure 4.6c and d are dominated by numerical diffusion. In the Lax-Wendroff algorithm which is second order, much less of the material originally in the square wave at $t = 0$ has diffused out at later times. Moreover, the amplitude errors are small enough that we can see the dispersion errors, which initially appear at the shortest wavelength and look similar to Gibbs errors. These appear as numerical ripples, predominantly in the wake of the square wave, and produce unphysical regions where $\rho < 0$. The pronounced undershoots and overshoots pose problems that are just as serious as those caused by excess numerical diffusion.

Figure 4.6f shows the same problem computed by the flux-corrected transport (FCT) algorithm (discussed in Chapter 8). FCT is a nonlinear algorithm designed to eliminate nonphysical ripples due to the Gibbs phenomenon and numerical dispersion. Nonlinear monotone methods, of which FCT is one example, are designed to minimize phase and amplitude errors and still maintain positivity and conservation. They represent years of efforts to improve solutions of continuity equations. Their development allowed major improvements in our ability to solve single and coupled sets of continuity equations, and thus to solve gas-dynamic problems accurately. The basic principle underpinning these algorithms is to impose additional, physically based constraints on the solutions. The series of algorithms developed have successively added more constraints, starting with positivity and monotonicity (*flux limiting)*, including analytic solutions of the properties of discontinuities (adding *Riemann solvers*), and even adding constraints to take advantage of foreknowledge of the direction of wave propagation (e.g., *upwinding*). At each stage, there are trade-offs that could be considered in the specific choice of algorithms for a specific problem. These trade-offs, and the resulting algorithms, are subjects of extended discussion in Chapters 8 and 9.

4–4.4. Phase and Amplitude Errors for Common Algorithms

The phase and amplitude properties for the five linear convection algorithms described above can be analyzed in terms of the amplification factor $A(k\,\Delta x)$ defined in equations (4–1.12) and (4–1.13). For a specific algorithm, A is found by substituting the test function

$$\rho_j^n = \rho^n(k)e^{ikj\,\delta x} \tag{4–4.14}$$

into the equation defining the particular algorithm, collecting the $\rho^n(k)$ and $\rho^{n-1}(k)$ terms, and then forming the ratio $A(k)$.

For diffusion equations, the imaginary part of the amplification factor is zero, which means that the phase shift for a given harmonic k is zero. The existence of a prevailing flow direction in convection ensures that A_I cannot be zero. By substituting equation (4–4.14) into equation (4–4.3) and taking the spatial derivative analytically, we obtain the exact theoretical result,

$$A^T(k\,\Delta x) = e^{-i\epsilon k\,\Delta x}. \tag{4–4.15}$$

This theoretical amplification factor A^T has unit magnitude, consistent with a profile that moves in space without growing or shrinking in size. We can also write the theoretically correct phase shift from equation (4–4.15), φ^T, as

$$\varphi^T = -\epsilon k\,\Delta x. \tag{4–4.16}$$

The amplification factors for the five linear algorithms considered above are determined by substituting the test function, equation (4–4.14), into equations (4–4.8), (4–4.9), and (4–4.13). The results are:

1. explicit solution, equation (4–4.8) with $\theta = 0$,

$$A^E(k\,\Delta x) = 1 - i\epsilon \sin(k\,\Delta x) \tag{4–4.17}$$

2. centered solution, equation (4–4.8) with $\theta = \frac{1}{2}$,

$$A^C(k\,\Delta x) = \frac{1 - i\frac{\epsilon}{2}\sin(k\,\Delta x)}{1 + i\frac{\epsilon}{2}\sin(k\,\Delta x)} \tag{4–4.18}$$

3. fully implicit solution, equation (4–4.8) with $\theta = 1$,

$$A^I(k\,\Delta x) = \frac{1}{1 + i\epsilon \sin(k\,\Delta x)} \tag{4–4.19}$$

4. first-order donor-cell algorithm,

$$A^{DC}(k\,\Delta x) = 1 - \epsilon + \epsilon\cos(k\,\Delta x) - i\epsilon \sin(k\,\Delta x) \tag{4–4.20}$$

5. second-order Lax-Wendroff algorithm,

$$A^{LW}(k\,\Delta x) = 1 - \epsilon^2(1 - \cos k\,\Delta x) - i\epsilon \sin(k\,\Delta x). \tag{4–4.21}$$

The amplification factor A for the explicit algorithm has unity in the denominator. Because the magnitude of the numerator is larger than unity, the algorithm is unstable. It is surprising how poor the result in Figure 4.6a is with only a small growth factor, $1.02 \approx \sqrt{1+\epsilon}$. Partially and fully implicit algorithms, which require solution of a tridiagonal matrix, generally have a denominator that is greater than unity, which counteracts the instability arising from the numerator.

Figure 4.7 shows the magnitude of the amplification factor, $|A(k)|$, and $\varphi_R(k)$, the ratio of the phase of each algorithm to the phase of the exact solution, as a function of $k\,\Delta x$

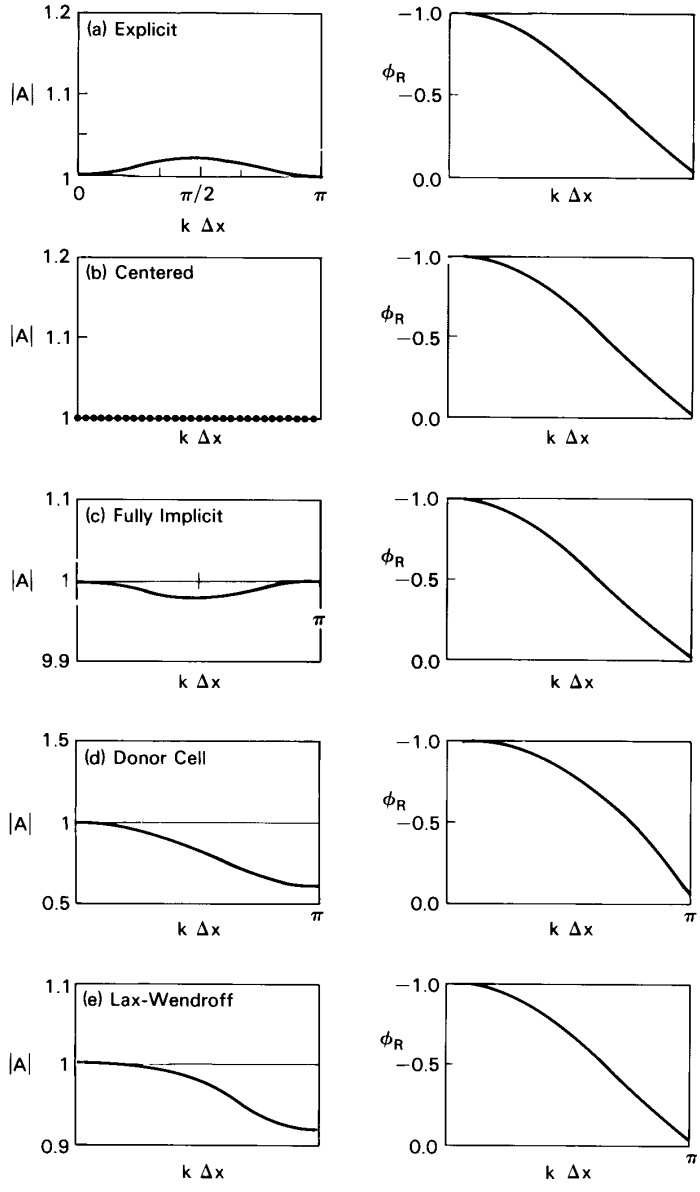

Figure 4.7. Amplitude and phase of the amplification factors for the five convective algorithms described in the text. (a) Explicit. (b) Centered. The dots on the horizontal axis mean that $|A| = 1$ everywhere. (c) Fully implicit. (d) Donor cell. (e) Lax-Wendroff.

for each of these five convection algorithms. Note the changes of scales on the graphs of $|A(k)|$. The horizontal axes in the graphs of $|A|$ are the exact solution. Also note how the errors in the phase increase with increasing values of $k\ \Delta x$.

As seen from Figures 4.2 and 4.3, the combination of phase errors and amplitude errors make it challenging to represent convection numerically on an Eulerian grid. Phase errors grow linearly in time and cause growing deviations from the correct solution. Amplitude

errors in convection, unlike those associated with diffusion, also result in growing rather than decreasing errors.

4–5. Waves and Oscillations

4–5.1. Waves from Coupling Continuity Equations

The fourth and last process listed in Table 4.1 is wave propagation. In the one-dimensional case, a linear wave in the scalar variable ρ is governed by the second-order wave equation

$$\frac{\partial^2 \rho}{\partial t^2} = v_w^2 \frac{\partial^2 \rho}{\partial x^2} \,. \tag{4–5.1}$$

Here v_w is the phase velocity of the wave, the velocity at which the wave profile moves.

Equation (4–5.1) does not appear explicitly in the full set of reactive-flow equations, but it arises from the interaction of two first-order equations through a coupling term (such as a pressure gradient or gravity). If $v(x, t)$ represents the fluid velocity, one possible set of first-order equations that generate a wave equation is

$$\frac{\partial \rho}{\partial t} = -\rho_o \frac{\partial v}{\partial x} \tag{4–5.2a}$$

$$\frac{\partial v}{\partial t} = -\frac{v_w^2}{\rho_o} \frac{\partial \rho}{\partial x}, \tag{4–5.2b}$$

where ρ_o is a constant with units of ρ.

Several kinds of waves are contained in the complete reactive-flow conservation equations, as may be shown by combining different terms to provide the two space and time derivatives. Sound waves couple the energy and momentum equations. Gravity waves arise from coupling the mass and momentum equations. Other types of waves are propagated by convection or driven by body forces that arise from convection in curved flows. Each kind of wave can exist over a spectrum of wavelengths, can propagate in a number of directions, and is generally accompanied by an energy flux.

All waves have some common physical and mathematical properties. These properties provide a partial basis for defining useful solution algorithms. One important property of most waves is energy conservation, and another is that the net displacement of fluid is usually quite small. The amplitudes of the variables ρ and v either change sign or fluctuate about a mean value as the wave oscillates. Thus positivity is not an important property in wave equations, though it is important in the underlying continuity equations.

The second-order derivatives in space make wave equations appear similar to the diffusion equation discussed in Section 4–3. Because the wave equation is also second order in time, it is reversible unless a physical dissipation process is present. Thus there should be no amplitude decrease in a linear wave equation algorithm. Implementing reversibility in the algorithm guarantees that modes do not damp when the algorithm is stable. In fact, the best algorithms are nearly unstable and can be easily driven unstable by small truncation errors, by additional uncentered terms, and by nonlinear effects that do not appear in the linear stability analysis.

Equation (4–5.1) or (4–5.2a,b) has oscillatory solutions in space and time of the form

$$\begin{aligned} \rho(x,t) &= \rho(k)e^{ikx}e^{-i\omega t} \\ v(x,t) &= v(k)e^{ikx}e^{-i\omega t}\ , \end{aligned} \tag{4–5.3}$$

where $\rho(k)$ and $v(k)$ are complex. Here k is the wavenumber in the x direction, a real quantity related to the wavelength of the mode by $\lambda = 2\pi/k$. The symbol ω denotes the angular frequency of the wave in the laboratory frame of reference. Inserting equation (4–5.3) into equation (4–5.2) yields the analytical solution,

$$\omega = v_w k. \tag{4–5.4}$$

In this theoretical case, the frequency of the wave increases linearly with the wavenumber k.

In the case of sound waves, velocity fields in three dimensions can be reduced to a single scalar wave equation by defining a velocity potential,

$$\mathbf{v} \equiv \nabla\phi, \tag{4–5.5}$$

and by assuming that the fluctuations in pressure, velocity, and density are small enough that the linear approximation is valid. In this case the variable ρ in equation (4–5.1) is replaced by ϕ, and v_w is the speed of sound $c_s \equiv \sqrt{(\gamma P/\rho)}$.

Only when v_w is constant can the wave equation be written as equation (4–5.1). More generally, v_w appears inside one or both of the spatial derivatives. This complication changes the character of the wave propagation appreciably in that the amplitude may now change when v_w varies in space and time. In this section, we analyze the special case in which the wave speed is independent of position and time and does not depend on the local values of $\rho(x,t)$ and $v(x,t)$. This situation is complicated enough. Both explicit and implicit reversible algorithms for waves are described below.

4–5.2. An Explicit Staggered Leapfrog Algorithm

Simple finite-difference formulas for equation (4–5.2) are

$$\frac{\rho_j^n - \rho_j^{n-1}}{\Delta t} = -v_w \left(\frac{v_{j+\frac{1}{2}}^{n-\frac{1}{2}} - v_{j-\frac{1}{2}}^{n-\frac{1}{2}}}{\Delta x} \right) \tag{4–5.6a}$$

$$\frac{\left(v_{j-\frac{1}{2}}^{n+\frac{1}{2}} - v_{j-\frac{1}{2}}^{n-\frac{1}{2}}\right)}{\Delta t} = -v_w \frac{\left(\rho_j^n - \rho_{j-1}^n\right)}{\Delta x}. \tag{4–5.6b}$$

The variable ρ is defined at the old and new times and at cell centers. The velocity variable v is defined at cell interfaces and at times that are midway between the times when the profile $\rho(x,t)$ is defined. Figure 4.8 shows an $x-t$ grid diagram for this algorithm. Because of the symmetry between the velocity and density grids, it is not entirely correct to think of one of the grids as forming computational cells or interfaces. These are called *staggered grids*.

The formulation of equation (4–5.6) is called an explicit, *staggered leapfrog algorithm* because the two variables jump forward over each other to advance the system in time, as shown in Figure 4.8. Because the new values of ρ depend only on the old values of ρ and

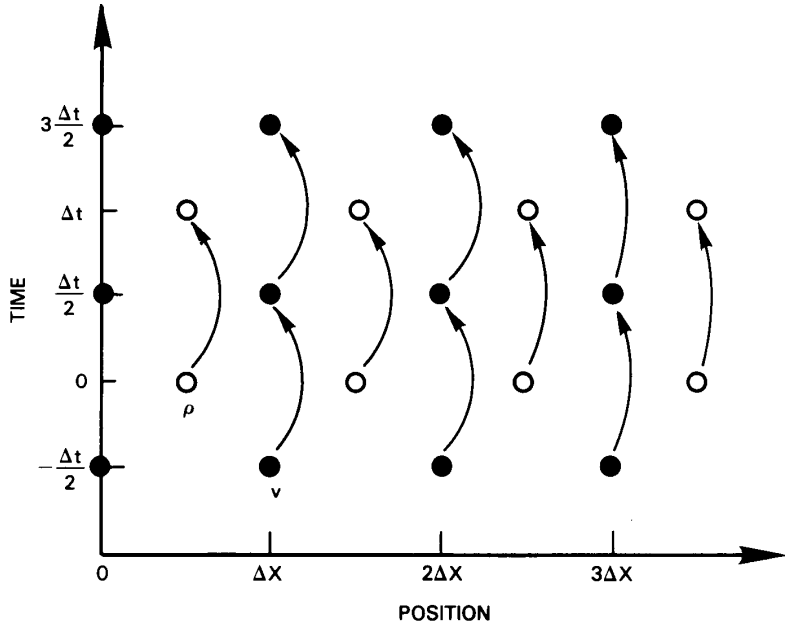

Figure 4.8. Diagram showing the relation between the variables ρ and v in the staggered leapfrog algorithm for solution of wave equations.

known values of v, not on each other, the algorithm is explicit and can be computed very efficiently. The large computational cost of solving implicit equations that couple the new values together is absent.

The staggered leapfrog algorithm is time reversible, as can be seen by inspecting equation (4–5.6) and Figure 4.8. At any step in the integration, the sign of the timestep can be changed and the exact values of the variables at previous steps can be recovered by running the algorithm backward. This property of the leapfrog algorithm is advantageous because it mirrors the reversibility property of the continuum wave in equations (4–5.1) and (4–5.2). Because it is reversible, the leapfrog algorithm maintains accurate mode amplitudes as long as it is numerically stable.

Using the definitions

$$\begin{aligned} x_j &= j\,\Delta x \\ x_{j+\frac{1}{2}} &= \left(j+\frac{1}{2}\right)\Delta x \\ t^n &= n\,\Delta t \\ t^{n+\frac{1}{2}} &= \left(n+\frac{1}{2}\right)\Delta t \end{aligned} \tag{4–5.7}$$

in equation (4–5.3) and substituting into equation (4–5.6) results in the following *numerical dispersion relation*

$$\sin\left(\frac{\omega_E\,\Delta t}{2}\right) = \epsilon \sin\left(\frac{k\,\Delta x}{2}\right). \tag{4–5.8}$$

Equation (4–5.8) gives the numerical wave frequency ω_E as a function of the wavenumber k and other parameters of the numerical system. Here the nondimensional timestep ϵ is

$$\epsilon \equiv \frac{v_w \, \Delta t}{\Delta x} \, . \tag{4–5.9}$$

Equation (4–5.8) can be solved formally for the numerical wave frequency,

$$\omega_E = \frac{2}{\Delta t} \sin^{-1} \left[\epsilon \sin \left(\frac{k \, \Delta x}{2} \right) \right]. \tag{4–5.10}$$

Equation (4–5.8) or (4–5.10) gives stability and accuracy bounds for the leapfrog algorithm. The algorithm converges because ω_E approaches the theoretical value ω in equation (4–5.4) as Δx approaches zero for fixed $\epsilon < 1$. As long as ϵ is less than unity, that is, as long as the real wave travels less than a cell per timestep, the magnitude of the argument of $\sin^{-1}$ in equation (4–5.8) can never exceed unity. This means that a real value of the numerical frequency ω_E can always be found with zero imaginary part, implying that the algorithm is numerically stable even though ω_E is quantitatively in error.

The numerical dispersion relation approaches the analytic solution given by equation (4–5.4) when the wavelength is many computational cells long, that is, when $k \, \Delta x \ll 1$. For finite values of $k \, \Delta x$, the numerical and theoretical dispersion relations differ. The exact solution gives one phase velocity for all waves, but each Fourier harmonic propagates at a different phase velocity in the finite-difference solution. Although the numerical waves are undamped, they propagate at the wrong phase velocity. As with a single continuity equation, this error is called dispersion.

The numerical solution becomes inaccurate when ϵ and $(k \, \Delta x)/2$ become appreciable fractions of unity. The worst dispersion errors occur at short wavelengths. There the sine function on the right-hand side of equation (4–5.8) causes the numerical dispersion relation to deviate from the theoretical expression equation (4–5.4). Inaccuracy has clearly set in by the time $k \, \Delta x$ approaches $\pi/4$, that is, eight computational cells per wavelength. Most of the modes in the system, however, have a shorter wavelength than eight cells. Thus the error in the computed frequency is appreciable even when the timestep (and hence ϵ) is very small. Because the dispersion comes from the finite-difference approximations to the spatial derivatives, it is not surprising that taking a smaller timestep cannot help.

Figure 4.9 shows the dispersion relations for the explicit leapfrog algorithm (E), the exact theoretical value (T), and an implicit algorithm (I). The curves show $(\omega \, \Delta t)/\epsilon \equiv (\omega \, \Delta x)/v_w$ as a function of $k \, \Delta x$. The theoretical curve is, by definition, the diagonal straight line.

Four different values of ϵ are shown in the four panels, spanning the transition from numerical stability to numerical instability for the explicit algorithm. At the stability boundary for E ($\epsilon = 1$), the theoretical and explicit dispersion relations agree exactly. This is shown by the diagonal line labeled T, E in Figure 4.9b. This exact result is obtained at the stability limit because the wave speed is exactly one spatial cell per timestep, a special situation which the leapfrog algorithm handles well.

Figure 4.9c and d shows cases where ϵ is increased above unity by taking a timestep larger than the explicit stability limit for short wavelength modes. For a given value of ϵ, there is a maximum value of $k \, \Delta x/2$ for which the algorithm is numerically unstable.

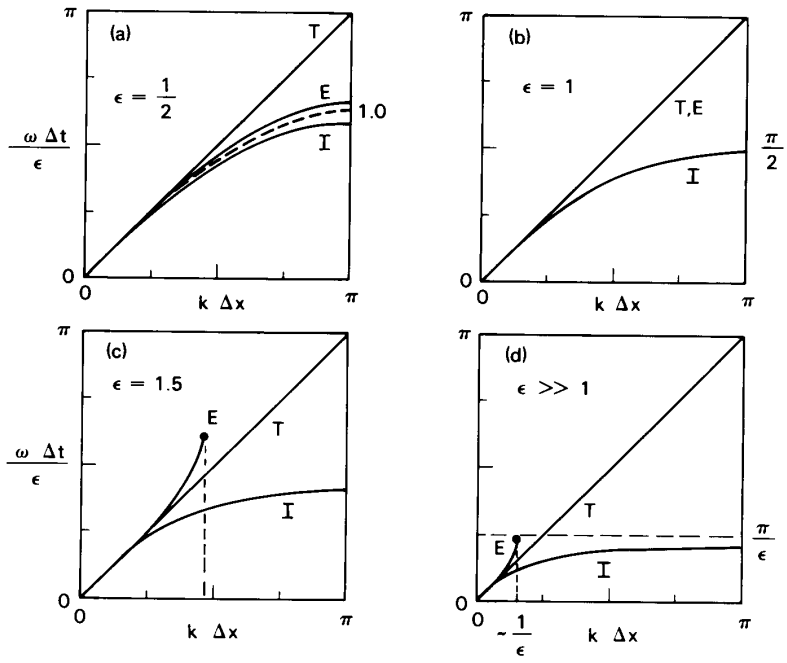

Figure 4.9. The dispersion relation for the solution of wave equations using an explicit (E) and an implicit (I) leapfrog algorithm, and the exact theoretical value (T). These are shown for four values of the parameter $\epsilon = v_w\ \Delta t/\Delta x$, (a) through (d).

In Figure 4.9c and d at $k\ \Delta x/2 = \sin^{-1}(1/\epsilon)$, the quantity $\epsilon \sin(k\ \Delta x/2)$ equals unity. At shorter wavelengths, the modes are numerically unstable, as indicated by the termination of curve E. As the timestep is increased, fewer modes are numerically stable, and those that are stable have longer wavelengths.

4–5.3. A Reversible Implicit Algorithm

Figure 4.9 also presents dispersion curves for an implicit algorithm (I) based on the formulas

$$\rho_j^n - \rho_j^{n-1} = -\frac{\Delta t\ v_w}{2}\left[\frac{\left(v_{j+\frac{1}{2}}^n - v_{j-\frac{1}{2}}^n\right)}{\Delta x} + \frac{\left(v_{j+\frac{1}{2}}^{n-1} - v_{j-\frac{1}{2}}^{n-1}\right)}{\Delta x}\right] \tag{4–5.11a}$$

$$v_{j+\frac{1}{2}}^n - v_{j+\frac{1}{2}}^{n-1} = -\frac{\Delta t\ v_w}{2}\left[\frac{\left(\rho_{j+1}^n - \rho_j^n\right)}{\Delta x} + \frac{\left(\rho_{j+1}^{n-1} - \rho_j^{n-1}\right)}{\Delta x}\right]. \tag{4–5.11b}$$

The grid here is staggered in space, but not in time. The new variables ρ and v are now determined simultaneously at a given timestep. This algorithm is reversible when the right sides of the equations are evaluated as a time-centered average of the spatial derivative. Reversibility, though computationally expensive, does guarantee that the individual mode amplitudes are preserved.

Because this implicit algorithm requires a matrix inversion, more computation is required at each timestep than with the explicit algorithm. To see the form of this matrix equation, we substitute equation (4–5.11b) into equation (4–5.11a) and eliminate the new values v. The result,

$$\rho_j^n - \frac{\epsilon^2}{4}\left(\rho_{j+1}^n - 2\rho_j^n + \rho_{j-1}^n\right) = \rho_j^{n-1} + \frac{\epsilon^2}{4}\left(\rho_{j+1}^{n-1} - 2\rho_j^{n-1} + \rho_{j-1}^{n-1}\right) - \frac{\Delta t\, v_w}{\Delta x}\left(v_{j+\frac{1}{2}}^{n-1} - v_{j-\frac{1}{2}}^{n-1}\right), \tag{4–5.12}$$

is a second-order finite-difference approximation to equation (4–5.1). The left side of equation (4–5.12) involves three adjacent values at the new time and is generally solved using a tridiagonal matrix algorithm. This extra work is similar to implicit solutions discussed for diffusion and convection equations. In two or three dimensions, the computational difficulties are far worse because the algebraic finite-difference equations are coupled in all directions. The tridiagonal equation, equation (4–5.12), becomes pentadiagonal in two dimensions, or septadiagonal in three dimensions.

Substituting the oscillatory trial solutions into equations (4–5.11) or equation (4–5.12) gives the following numerical dispersion relation for the reversible implicit algorithm,

$$\omega_I = \frac{2}{\Delta t}\tan^{-1}\left[\epsilon \sin\left(\frac{k\,\Delta x}{2}\right)\right]. \tag{4–5.13}$$

Equation (4–5.13) is the same as equation (4–5.10) except that an inverse tangent function replaces the inverse sine. This change ensures that the implicit method is always stable numerically because a real value of ω can always be found regardless of the values of $k\,\Delta x$ and ϵ. Figure 4.9 shows that ω_I always falls below both the theoretical value and also below the explicit value when the explicit algorithm is stable. Thus the explicit algorithm is again everywhere more accurate than the implicit algorithm, whenever the explicit algorithm is stable.

4–5.4. Reversibility, Stability, and Accuracy

Performing the actual calculations to verify that an algorithm is reversible is a crucial test of the computer program. Almost all programming errors, inconsistencies in definitions of variables, grid location errors, or incorrect finite-difference formulas destroy the exact reversibility property. One of the very best checks that can be made on the nominally reversible components of the model is to run the calculation for many timesteps, stop it, change the sign of Δt, and run the problem back to the initial conditions.

Just because an algorithm is fully reversible does not mean that it is necessarily accurate. As long as the explicit reversible algorithm is stable, it is more accurate than the corresponding implicit algorithm. Algorithms generally become inaccurate before the stability boundary is reached. As long as $\epsilon \leq 1$, that is, the wave moves less than a cell per timestep, we can always find a real value of ω for any choice of $k\ \Delta x$, and thus the solution is stable. When $\epsilon > 1$, then $\sin(\omega\ \Delta t/2) > 1$ for modes with $k\,\Delta x/2$ near $\pi/2$. These modes are unstable because complex values of ω are required to satisfy the numerical

dispersion relation. As a rule of thumb, it is usually courting disaster to run an algorithm with the longest stable timestep. Nonlinear effects, spatial variations of parameters, and variable cell sizes all tend to enforce a practical stability limit which is a factor of two or so more stringent than the mathematical limit.

When ϵ is very large, almost all of the waves travel more than a cell per timestep and thus are unstable if integrated explicitly. This is shown in Figure 4.9d. If $\epsilon > 1$, the maximum stable wavenumber for the explicit algorithm $k_{\max}$ satisfies

$$\frac{k_{\max}\,\Delta x}{2} \sim \frac{1}{\epsilon} > 1, \tag{4–5.14}$$

and the maximum value which the implicitly determined frequency ω_I can have is

$$\omega_{\max} = \frac{\pi}{\Delta t}. \tag{4–5.15}$$

For wavelengths that are unstable using an explicit algorithm, that is, for $k > k_{\max}$ from equation (4–5.14), the argument of the inverse tangent in the dispersion relation for the implicit algorithm, equation (4–5.13), is on the order of unity. Errors in the frequency of the waves, and hence the phase velocities of the modes, are thus also of order unity. In wave equations, as in convection and diffusion equations, stability conditions for the explicit method generally correspond to accuracy conditions for the stable, fully implicit methods. When the timestep is very large compared to what would be stable in an explicit algorithm, only the very longest waves in the system are properly calculated by the implicit algorithm. Thus you should be extremely careful using implicit algorithms. Just because an implicit calculation is stable does not mean that it is even approximately correct.

For diffusion equations, we did not have to consider the phases of the various modes. The amplitudes were the important quantities. In the case of waves, questions of accuracy focus on the phases. In the two algorithms discussed here, the mode amplitudes were held fixed by construction. The modes most in error, the short wavelengths, do not decay as they do in diffusion. The reversible, centered implicit algorithm, with timesteps much longer than the explicit stability condition requires, must be used cautiously. If some physical damping or diffusion is present in addition to the waves, the short wavelengths decay. There might not be a problem if the amplitudes become small enough before the growing phase errors become objectionable. If, however, waves are weakly damped, the short wavelengths may be troublesome enough to require the use of a Fourier or spectral method that minimizes both phase and amplitude errors, though Gibbs errors still require special consideration.

Asymptotic methods are much more difficult to formulate for waves than for local effects. An asymptotic method might be considered to solve the wave equation if it is possible to average over fast oscillations and extract quadratic effects which grow in time. The mean values of the oscillating variables themselves are zero.

4–6. Approaches to Coupling the Terms

Modeling reactive flows involves more difficulties than the idealized problems treated in this chapter. In particular, the models must correctly describe complicated interactions of

Table 4.5. Interactions and Their Effects

Interaction Processes	Some Effects Caused
Chemistry/Diffusion	Laminar flames, corrosion
Chemistry/Convection	Buoyant flames, turbulent burning
Diffusion/Convection	Molecular mixing in turbulence, Double-diffusion instabilities
Chemistry/Waves	Chemical-acoustic instability, Deflagration-to-detonation transition
Diffusion/Waves	Damping of high frequency sound waves
Convection/Waves	Shocks, turbulent noise generation

the basic chemical and physical processes in Table 4.1. Occasionally a problem reduces to solving coupled equations where all the terms are of the same form, such as when we solve the nonlinear ordinary differential equations of chemical kinetics. More generally, it is necessary to solve equations with terms of several different forms, representing different physical processes.

Many of the interesting physical effects and numerical difficulties arise from the two-process interactions listed in Table 4.5. These six interactions are a good starting point for considering how to couple algorithms representing several processes. If there are N processes to be coupled, there are $N(N-1)/2$ interactions among them that should be tested and calibrated separately. Small errors in simulating convection can cause large errors in the overall simulation. Consider, for example, a simulation of chemical reactions occurring behind a shock. The shock heats the combustible mixture, and the raised temperature and pressure initiate chemical reactions. In some mixtures, an error of even a few percent in the temperature calculated immediately behind a shock can cause an order-of-magnitude change in the very sensitive chemical-reaction rates. This can subsequently cause large changes in the time delay for appreciable energy release to begin. More generally, each of the interactions listed in Table 4.5 has regions of parameter space where one variable is very sensitive to small errors in the calculation of other variables.

Several approaches have evolved to solve sets of time-dependent, nonlinear, coupled partial differential equations. The major methods are the *global-implicit method*, also called the *block-implicit method*, and the *fractional-step method*, also called *timestep splitting* or *operator splitting*. Other approaches to coupling, such as *finite-element methods*, are also used. Each of these has a somewhat different philosophy, although they do have common elements. These approaches are often combined into hybrid algorithms in which different coupling techniques are used for different interactions. For general reading on different ways multiple time scales are treated, we recommend the book edited by Brackbill and Cohen (1985).

A global-implicit method solves the coupled set of equations using fully implicit finite-difference formulas. This method takes advantage of the superior stability properties of implicit methods, and is a direct generalization of the implicit algorithms presented earlier to the case where several types of terms are treated at once. An inherent problem with the approach is the lack of accuracy in solving particular types of terms. For example, a

global implicit method would use a linear finite-difference approximation to the entire set of coupled equations, part of which describe convective transport. As shown above, we might need to use a nonlinear monotone method for convective transport, and this would be precluded. In general, solutions using global-implicit methods require inverting large matrices at each timestep. They can also lead to a false sense of security about the accuracy of the results because they are stable for long timesteps.

In the fractional-step approach, also called timestep splitting or operator splitting, the individual types of processes are solved independently, and the changes resulting from the separate partial calculations are coupled (added) together. The basic idea of timestep splitting originated in the direction-splitting approach developed by Yanenko (1971) for solving multidimensional fluid dynamics equations. The extension to reactive flows involves treating the different processes, and sometimes the different directions of flow, separately and then "adding" the effects together. In the overall solution procedure, the processes and the interactions among them may be treated by a variety of different analytic, asymptotic, implicit, explicit, or other methods. An advantage of this approach is that it can avoid many costly matrix operations and allows the best method to be used for each type of term. Some of the disadvantages are that it requires monitoring during the calculation and thought as to how the coupling is best achieved. Further, the solutions are subject to all of the different types of numerical instabilities that can arise in each of the individual methods used to solve each type of term. Although there has been some substantial progress and verification of timestep splitting in the last ten years, the "rules" for doing this are still more of an art form than a science.

Throughout this book, we emphasize timestep splitting for the overall coupling approach because it allows more flexibility in the choice of which algorithms are used to solve each type of process. Individual groups of processes, however, may well benefit from an implicit approach. A more complete discussion of coupling methods is given in Chapter 11, along with specific examples of how to implement timestep splitting in reactive-flow codes.

REFERENCES

Anderson, J.D., Jr. 1995. *Computational fluid dynamics*. New York: McGraw-Hill.

Brackbill, J.U., and B.I. Cohen. 1985. *Multiple time scales*. New York: Academic.

De Jong, M.L. 1991. *Introduction to computational physics*. Reading, Mass.: Addison-Wesley.

DeVries, P.L. 1994. *A first course in computational physics*. New York: Wiley.

Garcia, A.L. 1994. *Numerical methods for physics*. Englewood Cliffs, N.J.: Prentice Hall.

Gould, H., and J. Tobochnik. 1996. *An introduction to computer simulation methods*. 2nd ed. Reading, Mass.: Addison-Wesley.

Hoffman, J.D. 1992. *Numerical methods for engineers and scientists*. New York: McGraw-Hill.

Koonin, S.E., and D.C. Meredith. 1990. *Computational physics*. Reading, Mass.: Addison-Wesley.

MacKeown, P.K., and D.J. Newman. 1987. *Computational techniques in physics*. Bristol, England: Adam Hilger.

Nakamura, S. 1991. *Applied numerical methods with software*. Englewood Cliffs, N.J.: Prentice Hall.

Potter, D. 1973. *Computational physics*. New York: Wiley.

Press, W.H., S.A. Teukolsky, W.T. Vetterling, B.P. Flannery, M. Metcalf. 1996a. *Numerical recipes in Fortran 77: The art of scientific computing*. Vol. 1, 2nd ed. New York: Cambridge.

———. 1996b. *Numerical recipes in Fortran 90: The art of parallel scientific computing*. vol. 2. 2nd ed. New York: Cambridge.

Roache, P.J. 1982. *Computational fluid dynamics*. Albuquerque, N. Mex.: Hermosa Publishers.

Tannehill, J.C., D.A. Anderson, and R.H. Pletcher. 1997. *Computational fluid mechanics and heat transfer*. Washington, D.C.: Taylor and Francis.

Thompson, W.J. 1992. *Computing for scientists and engineers*. New York: John Wiley & Sons.

Wong, S.S.M. 1992. *Computational methods in physics and engineering*. Englewood Cliffs, N.J.: Prentice Hall.

Yanenko, N.N. 1971. *The method of fractional steps*. New York: Springer-Verlag.

5

Ordinary Differential Equations: Reaction Mechanisms and Other Local Phenomena

Coupled sets of ordinary differential equations (ODEs) are used to describe the evolution of the interactions among chemical species as well as many other local processes. ODEs appear, for example, when spectral and other expansion methods are used to solve time-dependent partial differential equations. In these cases, spatial derivatives are converted to algebraic relationships leaving ODEs to be integrated in time. ODEs also describe the motions of projectiles and orbiting bodies, population dynamics, electrical circuits, local temperature equilibration, momentum interchange among phases in multiphase flows, the decomposition of radioactive material, and energy level and species conversion processes in atomic, molecular, and nuclear physics.

Algorithms for integrating ODEs were not originally derived by numerical analysts or applied mathematicians, but by scientists interested in solving specific sets of equations for their particular applications. Bashforth and Adams (1883), for example, developed a method for solving the equations describing capillary action. One of the first algorithms to cope with the difficulties of integrating stiff ODEs was suggested by Curtiss and Hirschfelder (1952) for chemical kinetics studies. Ten years after Curtiss and Hirschfelder identified the stiffness problem in ODEs, Dahlquist (1963) identified numerical instability as the cause of the difficulty and provided basic definitions and concepts that are still helpful in classifying and evaluating algorithms. The importance of the practical applications has spurred active research in developing and testing integration methods for solving coupled ODEs. Continued efforts of applied mathematicians have put the numerical solution of ODEs on a sounder theoretical basis and have provided insights into the constraints imposed by stability, convergence, and accuracy requirements.

For reactive-flow computations, the primary interests are either to solve the complicated sets of ODEs that describe a set of chemical reactions or to reduce these to simpler, more tractable sets of equations. Coupled, nonlinear, first-order ODEs for reactive systems were shown in equation (2–1.2). The part of the equations describing chemical kinetics was isolated in equation (2–2.22), which describes the production and loss of reacting species:

$$\frac{\partial n_i}{\partial t} = Q_i - n_i L_i, \qquad i = 1, \ldots, N_s. \tag{5–0.1}$$

Here the $\{n_i\}$ are the number densities of the reacting species, and $\{Q_i\}$ and $\{n_i L_i\}$ represent

the production and loss terms, respectively. The $\{Q_i\}$ and $\{L_i\}$ themselves are functions of the $\{n_i\}$ and provide the nonlinear coupling among the various species.

The notation used in this chapter differs from that used in standard books and papers on ODEs. When discussing the generic properties of ODEs, we describe the evolution of the quantity $y(t)$ and call the independent variable t, instead of x, to emphasize the variation with time in reactive flows. When discussing chemical kinetics problems, we call the dependent variables $\{n_i(t)\}$, the number densities of the reacting species at time t.

There are a number of excellent books and review articles on the solution of ODEs. Introductory material that describes the rudiments of integrating classical ODEs is given in texts such as Ralston (1978) and most of the computational physics and engineering books listed in the Prologue of this book. More advanced, specialized books and reviews classify and analyze methods for solving ODEs, often with an emphasis on methods for stiff equations (Gear 1971; Lambert 1973; Shampine, Watts, and Davenport 1976; Bui, Oppenheim, and Pratt 1984; May and Noye 1984; Hairer, Norsett, and Wanner 1987; Byrne and Hindmarsh 1987; Hairer and Wanner 1991). The review by May and Noye, which is aimed at the researcher whose problems require solutions of sets of ODEs, serves as a basis for portions of Sections 5–1 through 5–3.

Besides giving a general review of methods for solving stiff ODEs, this chapter also provides some practical advice and information that is useful for solving complicated sets of ODEs embedded in reactive-flow computations. This class of methods is generally different from those designed for stand-alone solutions of ODEs, as trade-offs in accuracy and computational speed must be considered. Section 5–4 covers a series of topics and is meant to provide useful practical information for solving ODEs in reactive flows. Section 5–5 describes approaches aimed at reducing the complexity of the reaction mechanism and perhaps eliminating stiffness entirely.

5–1. Definitions and Properties

5–1.1. The Initial Value Problem

A first-order nonlinear ODE can be written in the form

$$\frac{dy}{dt} = f(t, y), \qquad y(t_o) = y_o, \tag{5–1.1}$$

where the independent variable, t, increases from an initial value t_o. An analogous set of coupled ODEs can be written in vector form,

$$\frac{d\mathbf{y}}{dt} = \mathbf{f}(t, \mathbf{y}), \qquad \mathbf{y}(t_o) = \mathbf{y}_o. \tag{5–1.2}$$

These are called initial value problems. Given the initial values $\mathbf{y}_o$, we want to integrate this system forward in time to determine the values of $\mathbf{y}(t)$ at later times. Equations of the form

$$\frac{d\mathbf{y}}{dt} = \mathbf{f}(\mathbf{y}), \qquad \mathbf{y}(t_o) = \mathbf{y}_o, \tag{5–1.3}$$

without the functional dependence on t appearing on the right hand side, are called *autonomous* systems of ODEs. Any system governed by equation (5–1.2) can be put into the autonomous form by adding one more equation for t to the system. The general solutions of equation (5–1.1) or equation (5–1.2) are families of curves. The initial conditions $y(t_o) = y_o$ single out particular solutions in each family.

One approach to solving sets of ODEs computationally is to represent the variations of a function as expansions of basis functions. The vector $\mathbf{y}(t)$ is expanded in a series of functions of t. The coefficients of each term in the series are chosen so that the sum of the terms representing $\mathbf{y}(t)$ satisfies the initial conditions, $\mathbf{y}_o$. If the expansion function coefficients are suitably chosen, evaluating these expansions at subsequent values of t gives an approximation to the solution. A common expansion method uses a Taylor series for determining $\mathbf{y}(t + \Delta t)$ in terms of $\mathbf{y}(t)$. This approach does not work well for stiff equations. Difficulties arise because the expansions can be extrapolated forward only a finite time before they become hopelessly inaccurate.

In this chapter we concentrate on methods which divide the interval of time t over which we wish to solve the ODEs into a discrete set of values $\{t^n\}$, where $n = 0, 1, \ldots, N$. The intervals, or timesteps, between the various values t^n are denoted as $\{h^n\}$, where

$$h^n \equiv t^n - t^{n-1}. \tag{5–1.4}$$

The exact solution to equation (5–1.1) at the nth time t^n is denoted by $y(t^n)$. The solution obtained though a finite-difference method is denoted y^n. Given y^n, we need to know how to determine the value at the next step, y^{n+1}, as accurately and efficiently as possible.

5–1.2. Accuracy, Convergence, and Stability

We can express a scalar ODE algorithm as

$$y^{n+1} = y^n + h^n \mathcal{F}\left(h^n,\ \{y^i\},\ \left\{ \left.\frac{dy}{dt}\right|^i \right\}, \ldots\right), \tag{5–1.5}$$

where $i = 0, 1, \ldots, n+1$ indicates the various discrete steps at which the solution has been calculated. The numerical algorithm which advances y^n is embodied in the function $\mathcal{F}$. The most useful numerical schemes depend on values at only a few previous steps, that is, $i = n, n-1, n-2, \ldots$.

Different kinds of errors appear during this procedure, several of which are illustrated in Figure 5.1. The upper curve in the figure is the exact solution $y(t)$ as a function of t in the interval t^n to t^{n+1}. The values of $y(t^n)$ and $y(t^{n+1})$ are the correct values for the solutions at t^n and t^{n+1} using y^o, the exact initial conditions specified at a number of steps earlier than t^o. The lower solid curve, $y^*(t)$, is the exact solution of the equations obtained using y^n, the numerically calculated value at t^n, as the initial condition for subsequent integration. Because y^n can deviate appreciably from the correct solution $y(t^n)$, systematic error will appear in the solution $y^*(t^{n+1})$, as shown in the figure. The dashed line connecting y^n and y^{n+1} is the numerical solution obtained from equation (5–1.5) using an appropriate function $\mathcal{F}$. This numerical solution also passes through the correct initial condition y^o because the numerical solution was initialized to that value and integrated to give y^{n+1}.

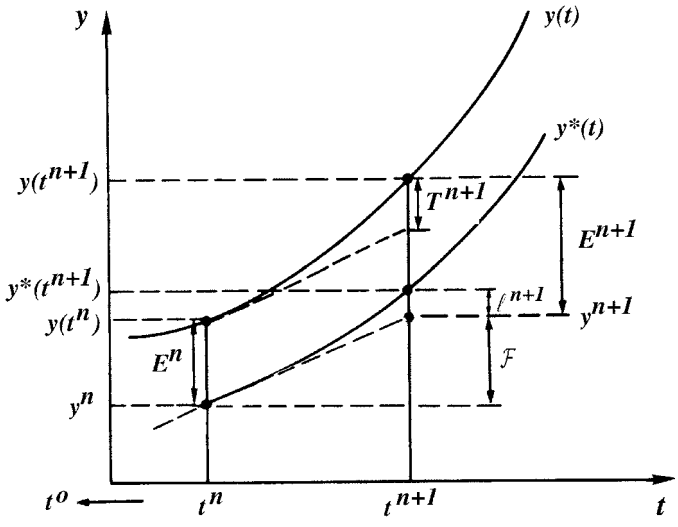

Figure 5.1. Types of errors in a finite-difference timestep. The exact solution, found from integrating from the starting condition at $y(t^o)$, is denoted $y(t)$. The exact solution starting from the numerical value at $y(t^n)$ is denoted $y^*(t)$. The global truncation error is denoted E^n. The local truncation error is denoted T^n. The local error is denoted ℓ^n.

The quantity E^n in Figure 5.1 is the *global truncation error* at t^n, defined by

$$E^n \equiv y(t^n) - y^n, \tag{5–1.6}$$

which is the difference between the exact solution at t^n based on the original initial conditions and the current numerical approximation. At t^{n+1} the global truncation error is E^{n+1}, as we show in the figure.

A *local truncation error*, T^{n+1}, is defined by

$$\begin{aligned} T^{n+1} &\equiv y(t^{n+1}) - y(t^n) - h^n \mathcal{F} \\ &= E^{n+1} - E^n. \end{aligned} \tag{5–1.7}$$

This is an estimate of the amount by which the approximation, equation (5–1.5), deviates from the analytic solution $y(t)$. The global truncation error is the sum of the local truncation errors made at each step.

Finally, there is a *local error*, ℓ^n, which is the error made in one step using a particular algorithm. When the values of the dependent variable at the previous step are used as initial conditions,

$$\ell^{n+1} = y^*(t^{n+1}) - y^{n+1}. \tag{5–1.8}$$

The local truncation error T^n is approximately equal to the local error ℓ^n when $y(t) \approx y^*(t)$. Although the exact solution is unavailable in practice, these different errors are important because they guide our thinking about errors and error propagation during the integration. In many situations, the fact that error can accumulate systematically over many steps is far more important than the error at any particular step.

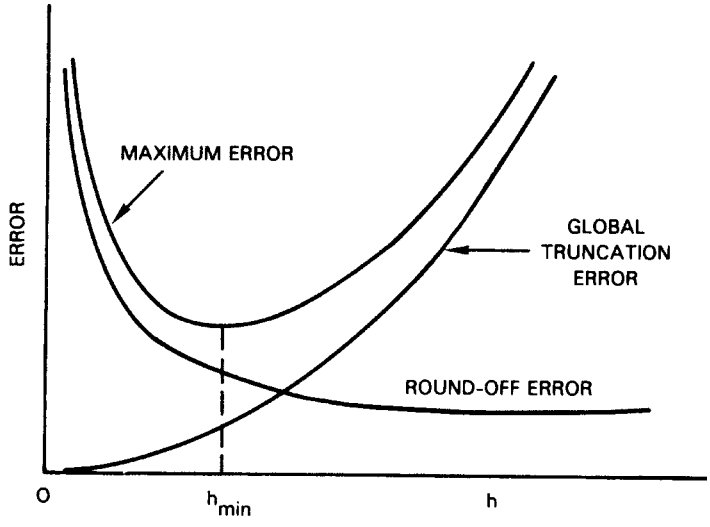

Figure 5.2. Schematic diagram showing error due to round-off and truncation as a function of timestep.

Errors are also introduced by machine-dependent numerical round-off. The global truncation error decreases as h decreases for a fixed integration interval, but the maximum round-off error increases. This is illustrated in Figure 5.2. Round-off error can contribute significantly to the global error when very short timesteps must be used. Thus there is an optimal stepsize, shown as $h^{\min}$ in Figure 5.2, where truncation and round-off errors are comparable.

The *order of accuracy* of a method is defined by expressing the measure of error in powers of the stepsize h. If the global truncation error behaves as $\mathcal{O}(h^p)$ for $p > 0$, the method is said to be pth order. In general, a pth order method has a local truncation error that scales as $\mathcal{O}(h^{p+1})$. The global truncation error scales as h^{-1} of the local truncation error when the errors accumulate proportionally to the number of steps, which is often the case. Higher-order methods usually converge faster and are more accurate. There are, however, many problems for which a lower-order method, perhaps with a smaller stepsize, is the best approach. A lower-order method, whose local truncation error changes sign and tends to cancel from step to step, may be more accurate globally than a high-order method whose truncation error accumulates.

Stability and convergence are both concerned with the behavior of the computed solution in the limit as the stepsize h goes to zero. A method *converges* if

$$y^n \to y(t^n), \qquad t \in [t^o, b]$$
$$\text{as } h \to 0 \quad \text{and} \quad y(t^o) \to y^o.$$

A method is said to be *unstable* if an error introduced at some stage in the calculation becomes unbounded, that is, the overall global error continually increases. Note that the numerical solution can be unstable yet remain relatively accurate if the real solution also grows indefinitely at a faster rate.

A method can also be stable, but not converge. For example,

$$y^n = y^{n-1}, \qquad n = 1, 2, \ldots$$

does not converge unless $\mathcal{F} = 0$, but it is stable. Stability does not require convergence, but convergence requires stability.

One reasonable way to control global error is to control the local error at each step. In general, stepsize control is the mechanism used to keep local errors acceptably small. Another strategy is to vary the order or even the type of method during the course of the integration depending on the nature of $\mathcal{F}(t, y)$.

During relatively short but important periods in the evolution of many reactive flows, fluctuations introduced to the fluid dynamics, transport, chemistry, or some combination of these, grow rapidly. Usually these periods of unstable growth are short because most physical instabilities disrupt the system that drives them. Nevertheless, understanding a numerical simulation during these periods of growth may require temporarily rethinking the meanings of stability and convergence. Because the relative errors are measured against a constantly growing scale, a weakly unstable algorithm may be accurate enough in a physically unstable environment. Conversely, in a system that is rapidly decaying to an equilibrium, a convergent algorithm may appear unstable because of increasing relative errors, even though the absolute errors are getting smaller.

In Chapter 4 we introduced stability concepts through a linear test problem given in equation (4–2.1). If a numerical method is unstable for such a simple problem, it must be judged unreliable at best for more complex problems. Most ODEs can be linearized locally by expanding in a Taylor series out to the first derivative. For small intervals about t^n, the values of $\mathbf{f}(\mathbf{y})$ will be close to $\mathbf{f}(\mathbf{y}^n)$, and thus

$$\mathbf{f}(\mathbf{y}) = \mathbf{f}(\mathbf{y}^n) + (\mathbf{y} - \mathbf{y}^n) \cdot \left.\frac{\partial \mathbf{f}}{\partial \mathbf{y}}\right|_{\mathbf{y}^n} + \text{ higher order terms.} \tag{5–1.9}$$

When this is substituted into equation (5–1.2), we obtain

$$\begin{aligned} \frac{d\mathbf{y}}{dt} &\approx \mathbf{y} \cdot \left.\frac{\partial \mathbf{f}}{\partial \mathbf{y}}\right|_{\mathbf{y}^n} + \left[\mathbf{f}(\mathbf{y}^n) - \mathbf{y}_n \cdot \left.\frac{\partial \mathbf{f}}{\partial \mathbf{y}}\right|_{\mathbf{y}^n}\right] \\ &\approx \mathbf{\Lambda} \cdot \mathbf{y} + \mathbf{g}(\mathbf{y}^n), \end{aligned} \tag{5–1.10}$$

which is the vector extension of the linear problem treated in Chapter 4. The term in square brackets defines $\mathbf{g}(\mathbf{y}^n)$, an additive constant vector. This term shifts the zero point of the dependent variables $\{\mathbf{y}\}$. By considering small enough intervals, the integral of the approximate equation (5–1.10), is arbitrarily close to the correct solution.

In this linear case, $\mathbf{\Lambda}$ is the $N \times N$ Jacobian matrix $\mathbf{J}$, whose elements are given by

$$J_{ij} \equiv \frac{\partial f_i}{\partial y_j}, \tag{5–1.11}$$

where N is the number of independent variables in the autonomous system of equations. Let $\{\lambda_i\}$ be the set of eigenvalues of $\mathbf{\Lambda}$ and let λ be the eigenvalue in the scalar case. When the real part of λ is negative, y approaches an equilibrium value y^∞ as t approaches $+\infty$. When the real part of λ is positive, the problem is said to be unstable, and y only approaches y^∞ as t approaches $-\infty$.

For the scalar case, or for each of the individual eigenvectors of the matrix problem given in equation (5–1.10), we can write

$$(y^{n+1} - y^{\infty}) = r(\lambda h)(y^n - y^{\infty}). \tag{5–1.12}$$

Here

$$y^{\infty} \equiv \frac{hg(y^n)}{(1 - r(\lambda h))}, \tag{5–1.13}$$

and r is a function of λh that depends on the particular method. Equation (5–1.12) describes the relaxation (or instability) of the linear solutions toward an equilibrium value y^{∞}. The limiting value y^{∞} is undefined when $r(\lambda h)$ is identically unity. The function $r(\lambda h)$ is the numerical approximation to the analytic exponential, that is,

$$r(\lambda h) = e^{\lambda h} + \mathcal{O}(h^{p+1}), \tag{5–1.14}$$

where p is the order of the method. The local and truncation errors are bounded and decay if

$$|r(\lambda h)| \leq 1. \tag{5–1.15}$$

Absolute stability is defined as

$$|r(\lambda h)| < 1. \tag{5–1.16}$$

The region in the complex λh plane where equation (5–1.16) is satisfied is called the *region of absolute stability.*

An algorithm for equation (5–1.1) or (5–1.2) is said to be *explicit* if the function $\mathcal{F}$ does not involve the variable y^{n+1}, and *implicit* if it does. In Chapter 4, we saw that implicit methods are usually more stable and slightly less accurate than stable, explicit methods and are more expensive to implement. Implicit methods generally require iteration, and each iteration step can be as costly as the explicit algorithm. When the solution is slowly varying, however, implicit methods allow long stepsizes that would make an explicit algorithm unstable. Thus they are particularly useful when the ODEs are stiff.

5–1.3. Stiff Ordinary Differential Equations

In practical computations, a system is stiff if the stepsize, based on cost or running time, is too large to give a stable, accurate answer. This statement describes both the minimum tolerable stepsize based on available computer time for a particular application and the solution method chosen.

Consider two coupled equations, perhaps in the form of equation (5–1.2), which have the solution

$$\begin{aligned} y_1(t) &= +e^{-2000t} + e^{-2t} + 1 \\ y_2(t) &= -e^{-2000t} + e^{-2t} + 1. \end{aligned} \tag{5–1.17}$$

There are two transients here. The fast one, $\exp(-2000t)$, decays by time $t = 0.01$. The slow one, $\exp(-2t)$, decays by $t = 10$, leaving the steady-state solution $y_1 = y_2 = 1$. An accurate numerical integration uses a small stepsize for the period $t < 0.01$ because the solutions are changing rapidly and the algorithm has to resolve that variation. After this fast transient, however, it would be useful to use a much larger stepsize.

Unfortunately, stability criteria based on the fastest relaxation rate must be considered even when these solution components themselves are not important. As a result, many methods require a small stepsize throughout the integration even when it appears that "nothing" is happening. Even though the numerical solution converges as $h \to 0$, h might have to be intolerably small. Often h must be so small for stability that round-off errors become critical. For example, $h \sim 5 \times 10^{-4}$ s might be required throughout the integration, so that it would take 20,000 steps to reach ten seconds of real time. The system is practically and mathematically stiff.

Stiffness occurs because there is a wide range of time scales in the solution. This leads to another more rigorous definition of stiffness (Lambert 1980), namely, that a set of equations of the form

$$\dot{\mathbf{y}} = \mathbf{J} \cdot \mathbf{y}, \tag{5–1.18}$$

where $\mathbf{J}$ is the Jacobian matrix, is stiff if

$$\begin{aligned} &(i) \quad \Re(\lambda_j) < 0, \qquad \text{for } j = 1, \ldots, N, \quad \text{and} \\ &(ii) \quad \frac{\max |\Re(\lambda_j)|}{\min |\Re(\lambda_j)|} \gg 1, \end{aligned} \tag{5–1.19}$$

where $\Re(\lambda)$ indicates the real part of λ. This definition of stiff does not apply when some of the independent modes of the system are growing exponentially. Equation (5–1.18) is mathematically stiff if $\mathbf{J}$ has at least one eigenvalue whose real part is negative and large compared to the time scales of variation displayed by the solution. A system is stiff if it contains transients which decay rapidly compared to the typical time scale of integration (Gear 1971; Curtis 1978).

Sets of coupled equations may be stiff, but a single equation can also be stiff (Gear 1971). Consider the equation

$$\frac{dy}{dt} = \lambda[y - F(t)] + \frac{dF(t)}{dt}, \tag{5–1.20}$$

where $\lambda < 0$, $|\lambda| \gg 1$, and $F(t)$ is a smooth, slowly varying function. The analytic solution is

$$y = [y^o - F(0)]e^{\lambda t} + F(t). \tag{5–1.21}$$

The λt exponent soon becomes negative enough so that the first term is negligibly small compared to the second. Nevertheless, λ controls the timestep. Compare this solution with equation (5–1.10). Here again, the constant term is not zero and is associated with a particular inhomogeneous solution to the equations, but the stability and stiffness are determined by the homogeneous solutions. Given a numerical method for solving equation (5–1.20), the local truncation error is determined by h and stability depends on the value of λh,

even though the solution, equation (5–1.21), does not. This is clearly unacceptable. We consider special methods for solving stiff equations in Section 5–3.

5–2. Overview of Classical Methods

Although common classical methods are not usually useful for treating the chemical kinetics terms in reactive-flow calculations, they provide the background needed to understand methods for stiff equations. In addition, the algorithms described here are useful in other contexts.

5–2.1. One-Step Methods

In a one-step method, the value of y^{n+1} at t^{n+1} is calculated knowing only h, t^n, and y^n. Information about the solution at only the most recent step is used. In two- or three-dimensional simulations, there are only a few fluid variables at each grid point but many species may be present. Each species density has to be integrated and stored at each grid point for each timestep. Thus for a large reactive-flow simulation, one-step methods are the methods of choice because they require a minimal amount of information to be stored in the computer memory. Even when computer memory is not a restriction, the nonlocal and fluid dynamic changes occurring in the simulation mean that extrapolations of species densities from previous timesteps are suspect if they are not updated to account for the fluid dynamic changes. Thus most large reactive-flow calculations use one-step methods.

A general one-step method can be written as

$$y^{n+1} = y^n + h\mathcal{F}(t^n, y^n, h), \tag{5–2.1}$$

from equation (5–1.5). The derivative approximation $\mathcal{F}$ can be evaluated either from a Taylor-series expansion about t^n,

$$y^{n+1} = y^n + h\frac{dy^n}{dt} + \frac{h^2}{2}\frac{d^2y^n}{dt^2} + \cdots, \tag{5–2.2}$$

or from approximations of the function $\mathcal{F}$ in the integral formula

$$y^{n+1} = y^n + \int_{t^n}^{t^{n+1}} \mathcal{F}(t, y(t))\, dt. \tag{5–2.3}$$

In the second case, approximations to $\mathcal{F}$ typically take the form of polynomials or exponentials.

The simplest one-step method is the explicit, first-order Taylor algorithm, some times called the *Euler method*,

$$y^{n+1} = y^n + hf(t^n, y^n). \tag{5–2.4}$$

Taylor-series algorithms for higher orders can be written analogously from equation (5–2.2). Usually the higher derivatives which appear are found from differentiating equations (5–1.1) through (5–1.3) analytically and substituting these expressions for the derivatives in the Taylor series.

Runge-Kutta methods are efficient, easily programmed, one-step algorithms that generally give higher-order accuracy at lower cost than Taylor-series methods. Their gains come from evaluating the function $f(t, y)$, at more than one point in the neighborhood of (t^n, y^n) instead of evaluating higher derivatives. The function $\mathcal{F}$ is expressed as a weighted average of first derivatives obtained numerically at points in the region (t^n, t^{n+1}).

The generic form for an R-stage Runge-Kutta algorithm is

$$k_r = f\left(t^n + ha_r,\, y^n + h\sum_{s=1}^{R} b_{rs}k_s\right), \qquad r = 1, 2, \ldots, R, \tag{5–2.5a}$$

where r labels the stage and

$$y^{n+1} = y^n + h\sum_{r=1}^{R} c_r k_r. \tag{5–2.5b}$$

The sequences of quantities $\{a_r\}$, $\{b_{rs}\}$, and $\{c_r\}$ are constants subject to a number of constraints, such as

$$\sum_{r=1}^{R} c_r \equiv 1. \tag{5–2.6}$$

Equations (5–2.5a,b) give explicit Runge-Kutta algorithms when all k values used as arguments to f are calculated at an earlier stage, that is, when $b_{rs} = 0$, for all $s \geq r$. Otherwise, equations (5–2.5a,b) give implicit Runge-Kutta algorithms. Many possible algorithms come from equation (5–2.5), as discussed in books by Gear (1971), Hairer, Norsett, and Wanner (1987), and Hairer and Wanner (1991). Because most of the ODEs that describe chemical reactions become stiff and these explicit methods are not applicable to stiff equations, only a few explicit Runge-Kutta algorithms are given in Table 5.1.

The two-stage modified Euler method is the basis for the second-order time integration in the Lax-Wendroff method and the flux-corrected transport algorithms described in Chapters 8 and 9. This method is straightforward to program and requires only two derivative evaluations per timestep. The most commonly used Runge-Kutta methods are the fourth-order explicit methods listed in Table 5.1. The last algorithm in the table is the Runge-Kutta-Gill algorithm (Gill 1951; May and Noye 1984) which we have found particularly useful. The computational form of the Runge-Kutta-Gill algorithm is given in Table 5.2.

In implicit Runge-Kutta methods, at least one of the coefficients b_{rs} is nonzero for $s \geq r$, so that at least one of the k_r's must be found implicitly. These implicit Runge-Kutta methods require iteration, but they are extremely accurate and stable. They are also applicable to stiff ODEs, as discussed in the next section.

5–2.2. Linear Multistep Methods

A k-step linear multistep method can be written in the general form

$$y^{n+k} = h\beta^k f^{n+k} + \sum_{j=0}^{k-1}(h\beta^j f^{n+j} - \alpha^j y^{n+j}), \tag{5–2.7}$$

Table 5.1. Runge-Kutta Integration Algorithms

Modified Euler method $\mathcal{O}(h^2)$

$$k_1 = f(t^n, y^n)$$
$$k_2 = f\left(t^n + \tfrac{1}{2}h, y^n + \tfrac{1}{2}hk_1\right)$$
$$y^{n+1} = y^n + hk_2$$

Improved Euler method $\mathcal{O}(h^2)$

$$k_1 = f(t^n, y^n)$$
$$k_2 = f(t^n + h, y^n + hk_1)$$
$$y^{n+1} = y^n + \tfrac{1}{2}h(k_1 + k_2)$$

Classical Runge-Kutta method $\mathcal{O}(h^4)$

$$k_1 = f(t^n, y^n)$$
$$k_2 = f\left(t^n + \tfrac{1}{2}h, y^n + \tfrac{1}{2}hk_1\right)$$
$$k_3 = f\left(t^n + \tfrac{1}{2}h, y^n + \tfrac{1}{2}hk_2\right)$$
$$k_4 = f(t^n + h, y^n + hk_3)$$
$$y^{n+1} = y^n + \tfrac{h}{6}[k_1 + 2k_2 + 2k_3 + k_4]$$

Runge-Kutta-Gill method $\mathcal{O}(h^4)$

$$k_1 = f(t^n, y^n)$$
$$k_2 = f\left(t^n + \tfrac{1}{2}h, y^n + \tfrac{1}{2}hk_1\right)$$
$$k_3 = f\left(t^n, +\tfrac{1}{2}h, y^n + \left(-\tfrac{1}{2} + \tfrac{1}{\sqrt{2}}\right)hk_1 + \left(1 - \tfrac{1}{\sqrt{2}}\right)hk_2\right)$$
$$k_4 = f\left(t^n + h, y^n - \tfrac{1}{\sqrt{2}}hk_2 + \left(1 + \tfrac{1}{\sqrt{2}}\right)hk_3\right)$$
$$y^{n+1} = y^n + \tfrac{h}{6}\left[k_1 + 2\left(1 - \tfrac{1}{\sqrt{2}}\right)k_2 + 2\left(1 + \tfrac{1}{\sqrt{2}}\right)k_3 + k_4\right]$$

where $\beta^k = 0$ is an explicit method and $\beta^k \neq 0$ is an implicit method. We typically need to know the values at a number of previous steps

$$(t^n, y^n),\ (t^{n-1}, y^{n-1}), \ldots, \tag{5–2.8}$$

at equally spaced intervals in h to compute y^{n+1}. Thus results must be stored for several steps back. Adams methods, the explicit Adams-Bashforth methods, and the implicit Adams-Moulton methods are all linear multistep methods. Examples of these methods up to $k = 4$ are given in Table 5.3. The $k = 1$ Adams-Bashforth method is the same as the first-order Euler method. The $k = 1$ Adams-Moulton method is called the *trapezoidal method* because it uses the trapezoidal quadrature formula. These *predictor-corrector methods* use a lower-order method to predict the answers until enough timesteps have accumulated to carry out the full multistep procedure. The coefficients $\{\alpha^j\}$ and $\{\beta^j\}$ may be derived in a number of different ways. One way, for example, is to use a Taylor-series expansion and match coefficients. A broader discussion of this topic is given by Lambert (1973), May and Noye (1984), or Gear (1971).

In practical applications, predictor-corrector methods compare favorably to Runge-Kutta methods. For equal computational effort, predictor-corrector methods can usually

Table 5.2. Computational Form of the Runge-Kutta-Gill Algorithm

Stage 1	given (t^n, y^n, q_4, and h)
	$k_1 = f\,(t^n, y^n)$
	$q_1 = q_4 + \frac{3}{2}\,(hk_1 - 2q_4) - \frac{1}{2}hk_1$ (q_4 from previous step)
	$w_1 = y^n + \frac{1}{2}\,(hk_1 - 2q_4)$
Stage 2	$k_2 = f\left(t^n + \frac{1}{2}h, w_1\right)$
	$q_2 = q_1 + 3\left(1 - \frac{1}{\sqrt{2}}\right)(hk_2 - q_1) - \left(1 - \frac{1}{\sqrt{2}}\right)hk_2$
	$w_2 = w_1 + \left(1 - \frac{1}{\sqrt{2}}\right)(hk_2 - q_1)$
Stage 3	$k_3 = f\left(t^n + \frac{1}{2}h, w_2\right)$
	$q_3 = q_2 + 3\left(1 + \frac{1}{\sqrt{2}}\right)(hk_3 - q_2) - \left(1 + \frac{1}{\sqrt{2}}\right)hk_3$
	$w_3 = w_2 + \left(1 + \frac{1}{\sqrt{2}}\right)(hk_3 - q_2)$
Stage 4	$k_4 = f\,(t^n + h, w_3)$
	$q_4 = q_3 + \frac{1}{2}\,(hk_4 - 2q_3) - \frac{1}{2}hk_4$
	$y^{n+1} = w_3 + \frac{1}{6}\,(hk_4 - 2q_3)$

be made more accurate, but Runge-Kutta methods generally require less storage and are *self-starting*, because they only require data at a single time level to begin the integration. In Runge-Kutta methods, it is easier to vary the stepsize. The Runge-Kutta methods are preferred for low accuracy requirements when the derivative evaluation is not expensive.

Linear multistep methods are generally not used to solve the ODE parts of large reactive-flow problems because they require storing values of **y** from more than one previous timestep. If timestep splitting is used to combine solutions of the various physical processes, if the chemistry must be subcycled on the fluid dynamic or diffusion timestep, and

Table 5.3. Adams Methods

Adams-Bashforth (explicit)

$$y^{n+1} = y^n + hf^n$$
$$y^{n+2} = y^{n+1} + \frac{h}{2}\left(3f^{n+1} - f^n\right)$$
$$y^{n+3} = y^{n+2} + \frac{h}{12}\left(23f^{n+2} - 16f^{n+1} + 5f^n\right)$$
$$y^{n+4} = y^{n+3} + \frac{h}{24}\left(55f^{n+3} - 59f^{n+2} + 37f^{n+1} - 9f^n\right)$$

Adams-Moulton (implicit)

$$y^{n+1} = y^n + \frac{h}{2}\left(f^{n+1} + f^n\right)$$
$$y^{n+2} = y^{n+1} + \frac{h}{12}\left(f^{n+2} + 8f^{n+1} - f^n\right)$$
$$y^{n+3} = y^{n+2} + \frac{h}{24}\left(9f^{n+3} + 19f^{n+2} - 5f^{n+1} + f^n\right)$$
$$y^{n+4} = y^{n+3} + \frac{h}{720}\left(251f^{n+4} + 646f^{n+3} - 264f^{n+2} - 106f^{n+1} - 19f^n\right)$$

Table 5.4. Euler-Romberg Coefficients

Iteration	Interval	Y_m^k				
	h_o	Y_o^o				
1	$h_o/2$	Y_o^1	Y_1^o			
2	$h_o/2^2$	Y_o^2	Y_1^1	Y_2^o		
3	$h_o/2^3$	Y_o^3	Y_1^2	Y_2^1	Y_3^o	
⋮	⋮	⋮	⋮	⋮	⋮	...

if the ODEs are not stiff, multistep methods can be used provided they are restarted at the beginning of each new chemistry interval.

5–2.3. Extrapolation Methods

In these methods, the equations are integrated several times by a lower-order method at successively smaller values of h. The full interval h is divided into N_o intervals of size h^o, N_1 intervals of size h^1, N_2 intervals of size h^2, and so on. Then a more accurate, higher-order solution is found by extrapolating the successive approximations to the zero timestep limit.

The Euler-Romberg method computes the solution to equation (5–1.2) by such an iterative procedure. Euler's method is applied several times over the same interval, but each integration is performed with twice as many steps using half the timestep of the previous integration. Then an extrapolation is made to $h = 0$ using the increasingly accurate integrations. The Euler-Romberg method is self-starting and rivals the best predictor-corrector methods for efficiency and accuracy. The choice of stepsize is fairly arbitrary, because the method halves the stepsize until the required accuracy is achieved.

Given the initial full stepsize h^o and initial conditions for the step (t^n, y^n), the method approximates y^{n+1} by constructing a table of values such as shown in Table 5.4. The procedure is started by computing an initial estimate for y^{n+1} (denoted Y_o^o in Table 5.4) using Euler's method and the full step h^o. By halving h^o and integrating two steps to reach h^o, values of Y_o^1 are generated.

The extrapolation part of the procedure is based on the linear interpolation formula

$$Y_m^k(h) = \frac{Y_{m-1}^k[h - h_{k+m}] - Y_{m-1}^{k+1}[h - h_k]}{h_k - h_{k+m}}. \tag{5–2.9}$$

This expression is linear in h in this example because the Euler method is only first-order accurate. It would have to be quadratic in h if the underlying method were second order. The best numerical approximation is found by extrapolating this equation to $h = 0$, giving

$$Y_m^k = \frac{2^m\, Y_{m-1}^{k+1} - Y_{m-1}^k}{2^m - 1}. \tag{5–2.10}$$

Equation (5–2.10) is used to fill each row of Table 5.4 after Y_o^k is computed using the Euler method with stepsize $h_o/2^k$. We continue iterating and generating the Y_m^k until the

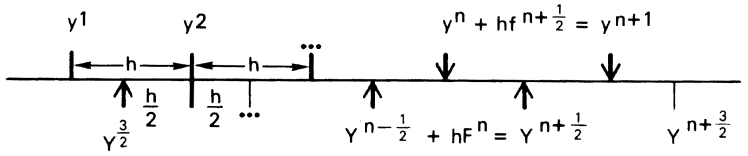

Figure 5.3. Time line diagram illustrating the leapfrog integration method on a staggered mesh.

extrapolated solution converges, as determined by the criterion

$$\left|Y_m^k - Y_m^{k-1}\right| \leq \epsilon, \tag{5–2.11}$$

where ϵ is a predetermined small quantity. A detailed description is given by McCalla (1967). Using a similar procedure with a method that is higher order than Euler's method produces extrapolations that converge more quickly.

Extrapolation methods are optimal when the evaluation of the derivative function is relatively expensive, the timestep for a given accuracy varies greatly in the course of the calculation, the equations are not stiff, or high accuracy is required. These methods are often based on the analysis by Neville (1934) that was later developed by Gragg (1965). One extension of the polynomial extrapolation methods shown above is a *rational extrapolation* method (Bulirsch and Stoer 1964, 1966) which corresponds to an extrapolation to $h = 0$ using rational functions of polynomials.

5–2.4. Leapfrog Integration Methods

The leapfrog algorithm is an explicit, second-order integration method. It is also reversible in time, a property that is particularly useful in integrating systems of ODEs that also are reversible. The main uses of the method are for dynamical systems represented as coupled, first-order differential equations rather than for chemical kinetic systems. Some waves propagate without growth or damping because they are reversible in time. Although phase errors can be present, the amplitude of each of the harmonics of the solution remains constant throughout a leapfrog integration.

The centered time integration of the leapfrog method ensures at least second-order accuracy. Reversibility is achieved by staggering the discretizations of the dynamical variables in time. The process is shown schematically as a time-line diagram in Figure 5.3. Two sets of variables are shown, $\mathbf{y}^n$ and $\mathbf{Y}^{n+\frac{1}{2}}$. The vertical marks in the figure above the time line indicate the discrete times t^{n-1}, t^n, t^{n+1}, ... at which the vector $\mathbf{y}^n$ is specified. Another vector, $\mathbf{Y}^{n+\frac{1}{2}}$, is indicated below the time line. The discretization of $\mathbf{Y}$ is staggered in time from that of $\mathbf{y}$ by a half step, $h/2$.

Consider the set of equations

$$\frac{d\mathbf{y}}{dt} = \mathbf{f}(\mathbf{Y}, t) \quad \text{and} \tag{5–2.12}$$

$$\frac{d\mathbf{Y}}{dt} = \mathbf{F}(\mathbf{y}, t). \tag{5–2.13}$$

Because the time derivative of $\mathbf{y}^n$ depends on the dependent variables $\mathbf{Y}^{n+\frac{1}{2}}$ but not $\mathbf{y}^n$,

and vice versa, the centered time derivative for each of the variables can always be evaluated explicitly using the most recently updated values of the other set of variables. For equations (5–2.12) and (5–2.13) the second-order leapfrog algorithm is

$$\mathbf{Y}^{n+\frac{1}{2}} = \mathbf{Y}^{n-\frac{1}{2}} + h\mathbf{F}(\mathbf{y}^n, t^n), \tag{5–2.14}$$

and

$$\mathbf{y}^{n+1} = \mathbf{y}^n + h\mathbf{f}\left(\mathbf{Y}^{n+\frac{1}{2}}, t^{n+\frac{1}{2}}\right), \tag{5–2.15}$$

where $t^{n+\frac{1}{2}} = t^n + \frac{h}{2}$. Because the algorithm is reversible, these two equations can be equally well solved for $\mathbf{y}^n$ and then $\mathbf{Y}^{n-\frac{1}{2}}$ by stepping through the algorithm in reverse order. Thus long integrations can be retraced to their initial conditions when computational round-off errors can be neglected.

Leapfrog algorithms can be generalized to include an implicit dependence of the variables,

$$\mathbf{Y}^{n+\frac{1}{2}} = \mathbf{Y}^{n-\frac{1}{2}} + h\mathbf{F}\left(\mathbf{y}^n, \frac{\mathbf{Y}^{n-\frac{1}{2}} + \mathbf{Y}^{n+\frac{1}{2}}}{2}, t^n\right), \tag{5–2.16}$$

$$\mathbf{y}^{n+1} = \mathbf{y}^n + h\mathbf{f}\left(\mathbf{Y}^{n+\frac{1}{2}}, \frac{\mathbf{y}^n + \mathbf{y}^{n+1}}{2}, t^{n+\frac{1}{2}}\right). \tag{5–2.17}$$

The form written in equations (5–2.16) and (5–2.17) is similar to the modified Euler method in that the derivative functions are evaluated at the center of the interval using the averaged dependent variables. As long as the implicit dependence on $\mathbf{Y}^{n+\frac{1}{2}}$ and $\mathbf{y}^{n+1}$ can be inverted analytically in the equations, the algorithm is reversible. If the implicit equations are solved by an iteration procedure, the reversibility of the algorithm is lost unless the iteration converges completely.

An alternate form of the implicit algorithm, based on the improved Euler method, may be easier to invert,

$$\mathbf{Y}^{n+\frac{1}{2}} - \frac{h}{2}\mathbf{F}\left(\mathbf{y}^n, \mathbf{Y}^{n+\frac{1}{2}}, t^n\right) = \mathbf{Y}^{n-\frac{1}{2}} + \frac{h}{2}\mathbf{F}\left(\mathbf{y}^n, \mathbf{Y}^{n-\frac{1}{2}}, t^n\right) \tag{5–2.18}$$

and

$$\mathbf{y}^{n+1} - \frac{h}{2}\mathbf{f}\left(\mathbf{Y}^{n+\frac{1}{2}}, \mathbf{y}^{n+1}, t^{n+\frac{1}{2}}\right) = \mathbf{y}^n + \frac{h}{2}\mathbf{f}\left(\mathbf{Y}^{n+\frac{1}{2}}, \mathbf{y}^n, t^{n+\frac{1}{2}}\right). \tag{5–2.19}$$

Reversible generalizations of this method to higher order are also possible. As with all of the classical techniques described so far, leapfrog methods break down for stiff equations. In principle, some form of implicit leapfrog algorithms could be applied to stiff systems. As in all implicit methods, an implicit leapfrog algorithm is more stable than an explicit leapfrog algorithm.

Leapfrog algorithms are particularly useful in orbit and particle dynamics calculations where the vector positions, $\mathbf{x} \equiv \mathbf{y}$, depend on the velocities, $\mathbf{v} \equiv \mathbf{Y}$, and the velocities depend on a force that is only a function of the instantaneous positions of the particles. This application has been described by Hockney (1965). An example of leapfrog algorithms in both classical particle dynamics calculations and semiclassical quantum mechanical calculations is given in Page et al. (1985).

5–3. Some Approaches to Solving Stiff Equations

Stiffness in ordinary differential equations is related to a basic computational problem: the presence of a wide range of time scales affecting the system dynamics. Mathematically, a system is stiff when the Jacobian matrix has eigenvalues whose magnitudes differ by a large ratio (see equation (5–1.18)). This means that at least two independent homogeneous solutions vary with time at very different rates.

Consider problems whose largest eigenvalues correspond to rapidly relaxing modes, so that their actual contribution to the evolution of the composite solution is negligible. In such problems, the timestep may have to be very small to accommodate the stability conditions of these modes. Although this case is mathematically stiff, an integration technique could use a long timestep if the high-frequency modes were treated stably. When we have such an algorithm, the problems are mathematically stiff but not practically stiff. There is no need to resolve the high-frequency modes because only their averaged effects are important. The stiff problems encountered in physics and reactive flows are very often of this type.

In some cases, the large eigenvalues of the Jacobian are positive so that the high-frequency modes grow rather than damp. These physically unstable situations lead to variations on short time scales that must be accurately resolved to determine the composite solution. Such problems are stiff both mathematically and practically. Fortunately, the stiffest or fastest modes usually do not grow. When they do, these violently unstable states saturate quickly and the growth becomes stabilized.

As a coupled nonlinear ODE system evolves, its Jacobian matrix also changes. This leads to changing patterns of stiffness in the system as the rapid, possibly unstable transients evolve. Once the period of fast growth is over, the stiffest modes in the system are damped in the vicinity of the new or quasistatic equilibrium. An exception occurs when high-frequency oscillations exist in the system for a long time. In these cases, the stiff modes are essentially undamped (the real parts of their eigenvalues are zero).

The numerical problems with stiffness cause us to seek methods that do not restrict the stepsize for stability reasons, and that can treat widely disparate scales in some reasonable manner. This need to solve stiff equations reliably has led to the development of nonclassical methods that address the questions of accuracy versus stability.

5–3.1. Stability and Stiff ODEs

When a set of equations is solved numerically, the stepsize h must be chosen according to two criteria:

1. The unstable modes with $\Re(\lambda_j) > 0$ must be represented accurately regardless of their intrinsic time scales, when they contribute significantly to the composite solution.
2. The values of $h\lambda_j$ must be within a region of absolute numerical stability for all j when $\Re(\lambda_j) < 0$.

If the second condition determines the stepsize h, the system is stiff. Below we introduce several aspects of stability in stiff systems of equations. This is important because relaxed stability conditions can allow for greater accuracy, particularly when there are growing modes.

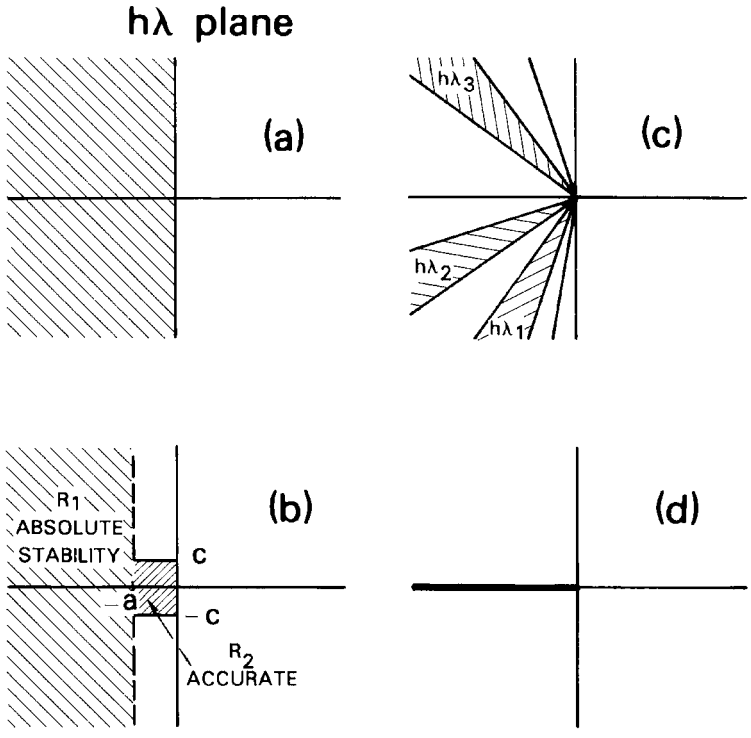

Figure 5.4. Regions of absolute stability in the $h\lambda$ plane for (a) A-stable, (b) stiffly stable, (c) A(α)-stable, (d) A_o-stable algorithms.

An implicit method such as the backward Euler method,

$$y^{n+1} = y^n + hf(y^{n+1}, t^{n+1}), \tag{5–3.1}$$

if applied to a test problem of the form

$$\frac{dy}{dt}(t) = \lambda y(t), \tag{5–3.2}$$

is stable if, for each step h, the error increases by a factor of the form $(1 - h\lambda)^{-1}$. Then if $\Re(\lambda)$ is less than zero, the error amplification is less than one. This is an example of *A-stability* (Dahlquist 1963). A numerical method is A-stable if, when applied to equation (5–3.2) with complex λ with a negative real part, the error due to the numerical approximation tends to zero as the number of timesteps goes to infinity.

This means that all physically decaying modes also decay numerically, but not necessarily at the correct rate. Though the numerical solution may be orders of magnitude larger than the physical solution, the error goes to zero because the whole numerical solution decays to zero. The A-stability condition says nothing about the accuracy of the physical modes that grow. For A-stability, the values of $h\lambda$ can fall anywhere in the negative half of the $h\lambda$ plane, as shown in the cross-hatched region of Figure 5.4a. In this case, and in

cases where all of the $\Re(\lambda_j)$ are less than zero, any value of $h > 0$ is within the region of absolute stability.

Dahlquist (1963) has shown that an A-stable multistep method can be at most second-order accurate. The trapezoidal and backward Euler methods, mentioned above, are examples of A-stable methods. For stiff ODEs, the trapezoidal method has the advantage of being precisely A-stable, but because it is only second order, it requires rather small stepsizes. Gear (1971) has summarized classes of methods that are A-stable. These include R-stage implicit Runge-Kutta methods of order $2R$ (defined in equation (5–2.5)) and methods that are implicit extensions of Taylor-series methods (Ehle 1969). For the lowest order, these reduce to the trapezoidal rule and the implicit midpoint rule, described below.

A method is *stiffly stable* (Gear 1969) if it is absolutely stable in the region R_1 of the complex $h\lambda$ plane for $\Re(\lambda) \leq -a$, and is accurate in the region R_2 for $-a < \Re(h\lambda) < b$, $-c < \Im(h\lambda) \leq c$, where $\Im$ indicates the imaginary part and a, b, and c are positive constants. This is shown as the cross-hatched region in Figure 5.4b, and is a slightly more relaxed criterion than A-stability. Gear has shown that there are k-step methods of order k which are stiffly stable for $k \leq 6$. These methods are used in the backward differentiation integrators based on the Gear method described in Section 5–3.2.

An even more relaxed criterion, *$A(\alpha)$-stability*, was proposed by Widlund (1967). A method is A(α)-stable, where α is in the range $(0, \pi/2)$, if all numerical approximations to $dy/dt = \lambda y$ converge to zero as the number of timesteps goes to infinity, with fixed h, for all $|\arg(-\lambda)| < \alpha$, $|\lambda| \neq 0$. This means that the region of absolute stability is a wedge or, for coupled ODEs, a series of wedges in the negative $h\lambda$ plane. This is illustrated in the cross-hatched regions of Figure 5.4c.

The least restrictive criterion, *A_o-stability* (Cryer 1973), applies when the region of absolute stability is the whole negative real axis. Figure 5.4 shows the regions of stability for the four criteria – A-stability, stiff-stability, A(α)-stability, and A_o-stability – given in order of descending restrictiveness. The less restrictive methods generally require shorter timesteps.

5–3.2. Implicit Methods for Stiff ODEs

An implicit multistep method for stiff equations differs from methods for nonstiff equations. Consider a linear multistep method described by a set of equations of the form of equation (5–2.7) and rewrite it as

$$\mathbf{y}^{n+k} = h\beta^k\, \mathbf{f}(t^{n+k}, \mathbf{y}^{n+k}) + \mathbf{B}^{k-1}, \tag{5–3.3}$$

where $\mathbf{B}^{k-1}$ is known at the kth step of the method,

$$\mathbf{B}^{k-1} = \sum_{j=0}^{k-1} (h\beta^j\, \mathbf{f}(t^{n+j}, \mathbf{y}^{n+j}) - \alpha^j \mathbf{y}^{n+j}). \tag{5–3.4}$$

When the equations are not stiff, we can use a direct iteration of the form

$$\mathbf{y}_{(m+1)}^{n+k} = h\beta^k\, \mathbf{f}\big(t^{n+k}, \mathbf{y}_{(m)}^{n+k}\big) + \mathbf{B}^{k-1}, \tag{5–3.5}$$

where subscript m is the iteration index. Evaluate $\mathbf{B}^{k-1}$, guess the value of $\mathbf{y}_o^{n+k}$, and then evaluate successive values of $\mathbf{y}^{n+k}$, that is, $\mathbf{y}_{(1)}^{n+k}$, $\mathbf{y}_{(2)}^{n+k}$, . . . , until the iteration converges.

This direct iteration approach does not work for stiff equations. It requires $h < 1/|J_{ij}|\beta^k$ for convergence because

$$|J_{ij}| = \left\| \frac{\partial f_i}{\partial y_j} \right\| \geq \max|\lambda_j| \tag{5–3.6}$$

would be very large, and therefore h would have to be very small.

This stiffness problem is handled by using Newton's method to solve equation (5–3.3). To solve the equation

$$\mathbf{F}(\mathbf{y}) = \mathbf{0} \tag{5–3.7}$$

using Newton's method, we have

$$\mathbf{y}_{(m+1)} = \mathbf{y}_{(m)} - \mathbf{F}\big(\mathbf{y}_{(m)}\big) \cdot \mathbf{J}^{-1}\big(\mathbf{y}_{(m)}\big), \qquad m = 0, 1, \ldots . \tag{5–3.8}$$

This gives the matrix equation

$$\mathbf{y}_{(m+1)}^{n+k} = \mathbf{y}_{(m)}^{n+k} - \big[\mathbf{y}_{(m)}^{n+k} - h\beta^k\, \mathbf{f}\big(t^{n+k}, \mathbf{y}_{(m)}^{n+k}\big)\big] \cdot \left[\mathbf{I} - h\beta^k\, \frac{\partial \mathbf{f}}{\partial \mathbf{y}}\big(t^{n+k}, \mathbf{y}_{(m)}^{n+k}\big)\right]^{-1}, \qquad m = 0, 1, 2, \ldots , \tag{5–3.9}$$

where $\mathbf{I}$ is the unit matrix. The iteration requires a good initial estimate of $\mathbf{y}_{(o)}^{n+k}$. Generally the equation is solved in the form where equation (5–3.9) is multiplied through by the last term on the left side. This is done to take advantage of the sparseness of the Jacobian matrix.

5–3.3. Useful Methods for Solving of Stiff ODEs

We now introduce a few of the more commonly used algorithms for solving stiff ODEs. Many others are covered, for example, by Hairer and Wanner (1991).

Backward Differentiation Formulas

These methods are the most common methods for solving systems of stiff equations and were described in detail by Gear (1971). They are linear multistep methods written in the form of equation (5–2.7) with $\beta^j = 0$,

$$y^{n+k} = h\beta^k f^{n+k} - \sum_{j=0}^{k-1} \alpha^j y^{n+j}, \tag{5–3.10}$$

where $\alpha^o \neq 0$ and $\beta^k \neq 0$, and the order is equal to the step number, k. For orders one through six these methods are stiffly stable. The first-order method is the backward Euler method. This method, as well as the second- and third-order methods, are absolutely stable in the right half of the complex $h\lambda$ plane, close to the origin (see Section 5–2.2 and Figure 5.4). The higher-order methods are not absolutely stable in a region of the left hand

Table 5.5. Coefficients for Backward Differentiation Methods

k	β_k	α_o	α_o	α_o	α_o	α_o	α_o	α_o
1	1	-1	1					
2	$\frac{2}{3}$	$\frac{1}{3}$	$-\frac{4}{3}$	1				
3	$\frac{6}{11}$	$-\frac{2}{11}$	$\frac{9}{11}$	$-\frac{18}{11}$	1			
4	$\frac{12}{25}$	$\frac{3}{25}$	$-\frac{16}{25}$	$\frac{36}{25}$	$-\frac{48}{25}$	1		
5	$\frac{60}{137}$	$-\frac{12}{137}$	$\frac{75}{137}$	$-\frac{200}{137}$	$\frac{300}{137}$	$-\frac{300}{137}$	1	
6	$\frac{600}{147}$	$\frac{10}{147}$	$-\frac{72}{147}$	$\frac{225}{147}$	$-\frac{400}{147}$	$\frac{450}{147}$	$-\frac{360}{147}$	1

plane near the imaginary axis and may give poor results for a system with an eigenvalue near the imaginary axis. In that case they either damp a solution that should be increasing in value, or they allow a solution to grow when it should decay. Table 5.5 gives values of the parameters for these formulas.

Exponential Methods

Implicit methods for stiff equations (see, for example, May and Noye [1984]) can be derived by *curve fitting* (Lambert 1973). In this process, a particular form of interpolating function is adopted with free parameters. These are determined by requiring the interpolant to satisfy certain conditions on the approximate solutions and their derivatives. For example, we can choose the two-parameter polynomial function

$$I(t) = A + Bt \tag{5–3.11}$$

as the interpolant in time, and require that $I(t)$ satisfy the constraints

$$I(0) = y^n, \qquad I'(0) = f^n, \qquad I(h) = y^{n+1} \tag{5–3.12}$$

on the interval $[0, h] = [t^n, t^{n+1}]$. This results in the Euler approximation

$$y^{n+1} = y^n + hf^n. \tag{5–3.13}$$

Other constraints and other forms of the interpolants result in different types of explicit and implicit methods.

One possibility is to use exponential functions to approximate the solution in the equilibration regime of stiff ODEs describing chemical kinetics (Keneshea 1967; Liniger and Willoughby 1970; Brandon 1974; Babcock, Stutzman, and Brandon 1979; Pratt and Radhakrishnan 1984) and this idea has been extended to use exponential interpolants. This approach is based on the idea that the exact solutions of stiff linear ODEs behave like decaying exponential functions. Because exponentials are poorly approximated by polynomials when the stepsize is larger than the characteristic decay rate, using exponential approximations should allow considerably longer timesteps. For example, consider the three-parameter exponential interpolant

$$I(t) = A + Be^{Zt} \tag{5–3.14}$$

for which A, B, and Z must be determined. The particular conditions in equation (5–3.12) determine A and B in terms of Z, as

$$A = y^n - \frac{f^n}{Z} \qquad B = \frac{f^n}{Z}, \tag{5–3.15}$$

yielding

$$y^{n+1} = y^n + hf^n\left[\frac{e^{Zh} - 1}{Zh}\right]. \tag{5–3.16}$$

There are a number of ways to choose the parameter Z, for example,

$$\begin{aligned}
&\text{explicit: } Z = f'^n/f^n \\
&\text{explicit: } Z = \frac{1}{h^{n-1}} \ln\left(\frac{f^n}{f^{n-1}}\right) \\
&\text{implicit: } Z = \frac{1}{h} \ln\left(\frac{f^{n+1}}{f^n}\right) \\
&\text{implicit: } Z = \frac{1}{2}\left[\frac{f'^n}{f^n} + \frac{f'^{n+1}}{f^{n+1}}\right]
\end{aligned} \tag{5–3.17}$$

where f' is used to represent d^2y/dt^2.

Another useful approach to solving the equations implicitly is by exponentially fitting a trapezoid rule (Brandon 1974; Babcock et al. 1979),

$$y^{n+1} = y^n + h\{\theta f^{n+1} + (1 - \theta) f^n)\}, \tag{5–3.18}$$

where θ is an implicitness parameter written in terms of Z,

$$\theta = \frac{1}{Zh} + \frac{1}{1 - e^{Zh}}. \tag{5–3.19}$$

This exponential-fitted trapezoidal rule is A-stable. It is equivalent to polynomial interpolants of at least order two and as great as six to eight. Exponential methods can be at least comparable in speed and accuracy to the backwards differentiation methods for stiff ODEs (Radhakrishnan 1984).

Asymptotic Methods

In Chapter 4 we described an asymptotic method applied to a single ODE. Such methods have a decided advantage over explicit and even implicit methods because of greater accuracy at large timesteps. This approach to solving stiff ODEs has been developed by Young and Boris (1977) and Young (1979), whose method comes from a second-order asymptotic expansion. To understand this approach, rewrite equation (2–2.22) as

$$\frac{dy}{dt} = Q(t) - y(t)\,\tau(t)^{-1} = f(y, t). \tag{5–3.20}$$

The quantity $\tau(t)$, the reciprocal of the "loss" coefficient $L(t)$, is the characteristic relaxation time describing how quickly the single variable y reaches its equilibrium value.

For simplicity we have dropped the subscript i on y, or equivalently, the vector notation indicating a coupled set of equations. The following arguments apply equally well to a coupled set representing many reacting species.

In this asymptotic approach, stiff equations, characterized by a sufficiently small value of τ, are solved in the form

$$\frac{y^{n+1} - y^n}{h} = \frac{Q^{n+1} + Q^n}{2} - \left(\frac{y^{n+1} + y^n}{\tau^{n+1} + \tau^n} \right). \tag{5–3.21}$$

The numerical problem here is that Q^{n+1} and τ^{n+1} are also implicit functions of y^{n+1}. The formal solution of equation (5–3.21) for y^{n+1} is

$$y^{n+1} = \frac{y^n(\tau^{n+1} + \tau^n - h) + \frac{h}{2}(Q^{n+1} + Q^n)(\tau^{n+1} + \tau^n)}{(\tau^{n+1} + \tau^n + h)}. \tag{5–3.22}$$

We use equation (5–3.22) as a corrector formula, and use

$$y^1 = \frac{y^o(2\tau^o - h) + 2hQ^o\tau^o}{2\tau^o + h} \tag{5–3.23}$$

as the predictor.

This method produces the best results when the solution is slowly varying, the rapidly relaxing modes are nearly at their asymptotic state, and the time constants are prohibitively small. This occurs when both Q and y/τ are large and nearly equal. The method is stiffly stable and tends to damp out small oscillations caused by very short time constants. One problem with the asymptotic method is that it does not necessarily conserve y. Although conservation cannot be guaranteed everywhere, it can be used as a convenient check on accuracy. Various modified asymptotic approaches to assure exact conservation have been devised, but they are usually not very satisfactory. When exact conservation is crucial, this method should not be used.

Asymptotic methods are very fast and moderately accurate. Their primary advantages are: (1) they do not require evaluation or use of the Jacobian or any matrix operations; (2) they are self-starting, requiring only initial values from one timestep; (3) they can be used with easily implemented stepsize estimation techniques; and (4) they can be combined with a simple, explicit algorithm for equations that are not stiff, which permits parallel processing of the equations. These features make the asymptotic methods convenient to combine with algorithms for fluid convection and diffusion processes, as in the program CHEMEQ, discussed in Section 5–4.3.

The New Quasi-Steady-State Approach

The asymptotic method was the starting point used by (Mott 1999) and Mott, Oran, and van Leer (2000, n.d.) for a new quasi-steady-state approach. Mott's approach is more stable, faster, and more accurate than the asymptotic method. To understand this approach, write equation (5–3.20) as

$$\frac{dy}{dt} = q - py, \quad y(0) = y^0. \tag{5–3.24}$$

If p and q are constant, equation (5–3.24) has an exact solution given by

$$y(t) = y^0 e^{-pt} + \frac{q}{p}(1 - e^{-pt}). \tag{5–3.25}$$

This equation is the basis of the class of methods called *quasi-steady-state* or *QSS* methods (Jay et al. 1997; Verwer and Simpson 1995; Verwer and van Loon 1994). Now the idea is to use equation (5–2.25) to form a predictor-corrector method, in which a prediction using only initial values is followed by a correction based on the initial and predicted values.

Equation (5–3.25) can be evaluated for timestep $\Delta t = h$ by

$$y(h) = y^0 + \frac{h(q - py^0)}{1 + \alpha\, ph}, \tag{5–3.26}$$

where

$$\alpha(ph) \equiv \frac{1 - (1 - e^{-ph})/(ph)}{(1 - e^{-ph})}. \tag{5–3.27}$$

The key, as explained by Mott, is to use the correct value of α.

A predictor-corrector method that uses equation (5–3.26) is

$$y^p = y^0 + \frac{h(q^0 - p^0 y^0)}{1 + \alpha^0 p^0 h}, \tag{5–3.28}$$

$$y^c = y^0 + \frac{h(q^* - p^* y^*)}{1 + \alpha^* p^* h}. \tag{5–3.29}$$

Here the superscripts p and c indicate predicted and corrected values, respectively. The predictor uses the initial values of p, q, and y, but the asterisked variables q^*, p^*, y^*, and α^* can be based on initial and predicted values. Mott gives various recipes for how to select these values optimally. In particular, one selection is

$$p^* = \frac{1}{2}(p^0 + p^p), \tag{5–3.30}$$

$$\alpha^* = \alpha(p^* h), \quad \text{from equation (5–3.37)}, \tag{5–3.31}$$

$$q^* = \alpha^* q^p + (1 - \alpha^*) q^0, \text{ and} \tag{5–3.32}$$

$$y^c = y^0 + \frac{h(q^* - p^* y^0)}{1 + \alpha^* p^* h}. \tag{5–3.33}$$

These choices of system parameters give a method that is second-order accurate and A-stable for linear problems.

Implicit Extrapolation Methods

Because explicit extrapolation methods (see Section 5–3.3) do not solve stiff ODEs any better than other explicit methods, implicit extrapolation methods have been developed. The simplest of these is the implicit or backward Euler method given by equation (5–3.1),

written here in vector form,

$$\mathbf{y}^{n+1} = \mathbf{y}^n + h\,\mathbf{f}(\mathbf{y}^{n+1}). \tag{5–3.34}$$

Equation (5–3.34) is also the lowest-order ($k = 0$) backward differentiation formula above. Because this formula is implicit, it requires the iterative numerical solution of a system of algebraic equations. For stiff equations, we use a Newton-type iteration of the form

$$\begin{aligned}(\mathbf{I} - h\,\mathbf{J})\cdot\mathbf{y}^{n+1}_{(m+1)} &= \mathbf{y}^n + h\,\bar{\mathbf{f}}\left(\mathbf{y}^{n+1}_{(m)}\right) \qquad m = 0, 1, \ldots \\ \bar{\mathbf{f}}(\mathbf{y}) &= \mathbf{f}(\mathbf{y}) - \mathbf{J}\cdot\mathbf{y},\end{aligned} \tag{5–3.35}$$

and $\mathbf{J}$ is evaluated at $\{\mathbf{y}^n\}$. This method is stable on the whole left half-plane, which is desirable, but it is too stable, or *superstable*, on the right half-plane. This means that increasing analytical solutions may be approximated by decreasing numerical solutions.

Other basic algorithms for implicit extrapolation methods include the implicit trapezoidal method,

$$\mathbf{y}^{n+1} = \mathbf{y}^n + \frac{h}{2}(\mathbf{f}(\mathbf{y}^n) + \mathbf{f}(\mathbf{y}^{n+1})) \tag{5–3.36}$$

and the implicit midpoint rule (Dahlquist 1963; Dahlquist and Lindberg 1973; and Lindberg 1973),

$$\mathbf{y}^{n+1} = \mathbf{y}^n + h\,\mathbf{f}\left(\frac{\mathbf{y}^n + \mathbf{y}^{n+1}}{2}\right). \tag{5–3.37}$$

These methods have asymptotic expansions and are both A(α)-stable. Nonetheless, none of the higher-order implicit extrapolations generally work as well as equivalent-order backward differentiation methods.

A Low-Order, Single-Step Extrapolation Method

Consider the equation

$$\frac{d\mathbf{f}}{dt} = \mathbf{J}\,\mathbf{f} \tag{5–3.38}$$

that can be derived from equation (5–1.3) and the definition of $\mathbf{J}$, equation (5–1.11). This may be discretized as

$$\mathbf{f}^n - \mathbf{f}^o = h\,\mathbf{J}^o\mathbf{f}^n, \tag{5–3.39}$$

which is a curious form since, on the right side, the Jacobian matrix $\mathbf{J}$ is evaluated at the old time and the source terms $\mathbf{f}$ are evaluated at the new time. Equation (5–3.39) can be rewritten as

$$(\mathbf{I} - h\mathbf{J}^o)\mathbf{f}^n = \mathbf{f}^o. \tag{5–3.40}$$

The definition of the Jacobian then leads to the approximate relation

$$\mathbf{f}^n - \mathbf{f}^o \approx \mathbf{J}^o(\mathbf{y}^n - \mathbf{y}^o). \tag{5–3.41}$$

This prescribes a two-step procedure from which $\mathbf{f}^n$ is found from equation (5–3.40) and then $\mathbf{y}^n$ is found from equation (5–3.41).

Note that combining equations (5–3.40) and (5–3.41) gives the form

$$(\mathbf{I} - h\mathbf{J}^o)(\mathbf{y}^n - \mathbf{y}^o) = h\mathbf{f}^o \tag{5–3.42}$$

that can be derived from the backward Euler method given in equation (5–3.35) for the case in which no iterations are made and $\mathbf{J}$ is evaluated at the old time $\mathbf{J}^o$.

This method was described by Wagoner (1969) and implemented by Arnett and Truran (1969) to solve a very stiff set of equations describing thermonuclear reactions. The importance of this method is that it is extremely stable and, with the correct choice of timesteps, can be very accurate. It is single step and self-starting. Khokhlov (n.d.) has shown that it is formally equivalent to finding all nonzero eigenvalues and eigenvectors of $\mathbf{J}$, linearly transforming $\mathbf{y}$ and $\mathbf{f}$ into the coordinate system where $\mathbf{J}$ is diagonal, finding the solution in the new coordinate system, and then transforming back to the original coordinate system. The method is the basis of the solver YASS described in Section 5–4.3.

5–3.4. Summary: Some Important Issues

There has been a substantial effort recently to develop methods for the numerical solution of stiff ODEs. This effort is driven by a number of applications, but most strongly by the types of reaction kinetics problems that interest us here. Several important issues have emerged from the efforts to find efficient, accurate algorithms.

The first important issue, control of the stepsize, has given rise to many stepsize estimation techniques. Stepsize control is crucial in all methods except those that use constant stepsizes or stepsizes that are fixed fractions of those previously used. Ideally we should take the largest stepsize compatible with the required accuracy. For nonstiff equations, this is not too great a problem. For stiff equations, unless implicit or asymptotic methods are used, the stepsize requirement may be so small that the answer is dominated by round-off error.

Even when the method used to solve a set of stiff equations is A-stable, the solution may have large errors from unresolved gradients if the stepsize is large. One advantage of using a less stable method (e.g., one that is stiffly stable) is that it can be unstable for large stepsize, thus giving a warning that the stepsize is too large for accuracy.

The second issue is the use of the Jacobian matrix. It is expensive to solve a large matrix equation because inversion requires on the order of M^3 operations, where M is the size of the matrix. An efficient algorithm saves computer time by using the old Jacobian as long as possible. Recalculating the Jacobian is expensive, but an inefficient convergence rate is a signal to reevaluate the Jacobian and to decompose the new matrix.

The third issue concerns how much accuracy is needed. For reactive-flow problems, where the species reactions are coupled to fluid dynamic flow, the solutions of the rate equations almost never reach the equilibrium regime where cancellations exceed round-off accuracy. Inevitable, fluid-induced fluctuations in temperature and pressure ensure slight

Table 5.6. The Costs of Chemical Kinetics Algorithms

Mechanism	Reacting Species	Reactions
$H_2 - O_2$	≈9	≈50
$H_2 - O_2 - N_2$	≈15	≈100
$CH_4 - O_2$ (lean)	≈30	≈200
$C_2H_6 - O_2$ (lean)	≈30	≈200
$CH_4 - O_2 - N_2$	≈50	≈350
Octane – air (no NOx)	≈800	≈3000
Soot formation in $CH_4 - O_2$	≈100's	≈10,000's

deviations from equilibrium, and this generally invites the use of less expensive methods. This aspect of coupling fluid dynamics and chemical kinetics algorithms is discussed in Chapter 11.

5–4. Useful Information and Techniques

The remainder of the chapter has a more practical emphasis than the previous material. Here we focus on efficient, accurate approaches for solving coupled sets of ODEs, when the set of equations are integrated alone and also for ODE systems that are part of a reactive-flow calculation. These two types of applications often call for rather different numerical approaches.

In a reactive-flow computation, solution of the chemical kinetics is often the most expensive part of the calculation. Table 5.6 gives the number of species and reactions among them in several chemical reaction mechanisms. The smallest mechanism in the table describes hydrogen oxidation, with nine reacting species and about fifty reactions. This number increases dramatically for higher hydrocarbons, and can be enormous for a mechanism that describes the details of soot formation.

As Table 5.6 shows, there may be hundreds, even thousands, of interacting species in flame and detonation simulations. Thus, integrating the chemical kinetics equations may take an order of magnitude longer than solving the convective and diffusive transport terms. The computational cost is directly related to the number of species, the number of reactions among them, and the number of spatial cells in the computational representation. The cost also depends strongly on the particular form of the expressions for the production and loss terms. When these rates involve exponentials or fractional powers of the temperature, as is usually the case with chemical rates, they become even more expensive to evaluate.

The first efforts to deal with the problem were to develop numerical integration algorithms that are fast while being accurate *enough* for the system being considered. This led to the development of a number of optimized approaches for treating sets of stiff equations and hybrid techniques for sets of mixed stiff and nonstiff equations. These stiff and hybrid integration techniques are described in Section 5–4.3. These techniques have also led to many discussions of how to optimize these methods on different types of computers (e.g.,

Burrage 1995; Weber et al. 1996). The second line of attack, which arose when the sets of equations themselves became very large, is to find ways to reduce the chemical mechanism itself to a more manageable size. Again, a key idea in this approach is to maintain just enough accuracy in the variables of interest for each particular problem.

5–4.1. Solving Temperature Equations

The evolution equation for the temperature must be solved in conjunction with equation (5–0.1) when chemical energy is being released because most reaction rates are very temperature dependent. The temperature equation does not appear explicitly in the conservation equations, but can be derived from the energy equation by relating the internal energy to the temperature

$$\epsilon = \frac{f}{2}kT, \tag{5–4.1}$$

where f is the number of degrees of freedom. Every chemical reaction occurring at local thermodynamic equilibrium releases or consumes a known amount of energy. Therefore, the time-rate-of-change of T is controlled by both the rate equations for the evolution of the reaction species, and by changes in the background due to other physical processes (such as convection or radiation transport).

Alternatively, because of these constraints, we can use the expression

$$\epsilon = H(T, \{n_j\}) - P, \tag{5–4.2}$$

where here H represents the enthalpy function of the system. (In this chapter, we used lower-case h to represent the stepsize. To avoid confusion, upper-case H here represents the enthalpy.) If we know the value of the enthalpy, we can find an expression for $\partial T/\partial t$. The expression for the temperature derivative involves all of the $\{\partial n_j/\partial t\}$ derivatives as well as expressions of the form $\{\partial H_j/\partial t\}$ that may be expressed as powers of T. Solving for the temperature by this approach is usually expensive computationally.

The correct expression for the enthalpy also involves complicated sums over excited states of the molecules. When each of the species is in local thermodynamic equilibrium, the individual $H_j(T)$ can be evaluated as a function of temperature and fit to a polynomial expansion. This has been done in the JANAF tables (Stull and Prophet 1971) and in the work of Gordon and McBride (1976). These data have been updated substantially in the National Institute of Standards and Technology (NIST) JANAF Thermochemical Tables Database (available in electronic form as volumes 1–14 of the *Journal of Physical and Chemical Reference Data* published by the National Institute of Science and Technology, Gaithersburg, Maryland) and in the newer work by McBride and Gordon (1996).

By using tabulated values of the enthalpies $\{H_i(T)\}$ and the heats of formation $\{H_{oi}(T)\}$ (see Chapter 2), a tedious temperature integration can often be avoided completely. During the chemical reaction portions of each timestep, assume that the total internal energy of the system does not change but may be redistributed among chemical states. Then equation

(5–4.2) can be solved iteratively using a Newton-Raphson algorithm to give a temperature which is consistent with the new number densities being calculated. When incorporated in simulations, the solution converges in one or two iterations per timestep.

5–4.2. Sensitivity Analysis

Knowing the sensitivity of the behavior of a system of ODEs to uncertainties in its input parameters, such as chemical rates or initial conditions, is particularly useful for developing and using complicated reaction mechanisms. For example, the chemical rates are often semiquantitative formulas extracted from experiments or fundamental calculations. They can be strong functions of both temperature and pressure, and often their values are not well known. An estimate of the uncertainties in these parameters and how these uncertainties affect the reaction mechanism is an important part of determining and assessing the validity of the reaction model and formulating simplifed or *reduced* mechanisms.

Two general classes of methods have evolved for evaluating the effects of uncertainties. The *local* or *deterministic* methods produce information on how the uncertainty in one parameter – for example, a chemical reaction rate k_i – affects the calculated results of one of the dependent variables – for example, y_j. The *global* or *stochastic* methods consider the effects of simultaneously varying parameters over a range of values. The sensitivity measure that results from this global approach gives an average effect of the uncertainties.

We now describe some of the principles of local sensitivity methods, leaving details and discussion of global methods to be found in the references cited. A number of reviews have been written about sensitivity methods. We recommend those by Tilden et al. (1981) and Rabitz, Kramer, and Dacol (1983). More recent overviews are given in Radhakrishnan (1991) and Griffiths (1995).

Local Sensitivity Methods

Equation (5–1.1) may be rewritten as

$$\frac{d\mathbf{y}}{dt} = \mathbf{f}(\mathbf{y}, \mathbf{k}), \qquad \mathbf{y}(0) = \mathbf{y}_o, \tag{5–4.3}$$

where $\mathbf{y}(t)$ is the solution vector of N species concentrations, and $\mathbf{k}$ is a vector of M input parameters such as chemical rate constants and activation energies. This form of the equations emphasizes the functional dependence of $\mathbf{y}(t)$ on the parameters $\mathbf{k}$. There is usually an estimated value, $\widehat{k}_j$, and uncertainties associated with each value k_j.

If only small deviations of k_j are considered about $\widehat{k}_j$, a truncated Taylor series expansion can be used to estimate $y_i(t, \mathbf{k}) = y_i(t, \widehat{\mathbf{k}} + \Delta\mathbf{k})$,

$$y_i(t, \mathbf{k}) = y_i(t, \widehat{\mathbf{k}}) + \sum_{j=1}^{m} \frac{\partial y_i(t, \widehat{\mathbf{k}})}{\partial k_j} \Delta k_j. \tag{5–4.4}$$

The first-order sensitivity coefficient matrix is defined as

$$\beta_{ij} \equiv \left. \frac{\partial y_i}{\partial k_j} \right|_{\widehat{\mathbf{k}}}. \tag{5–4.5}$$

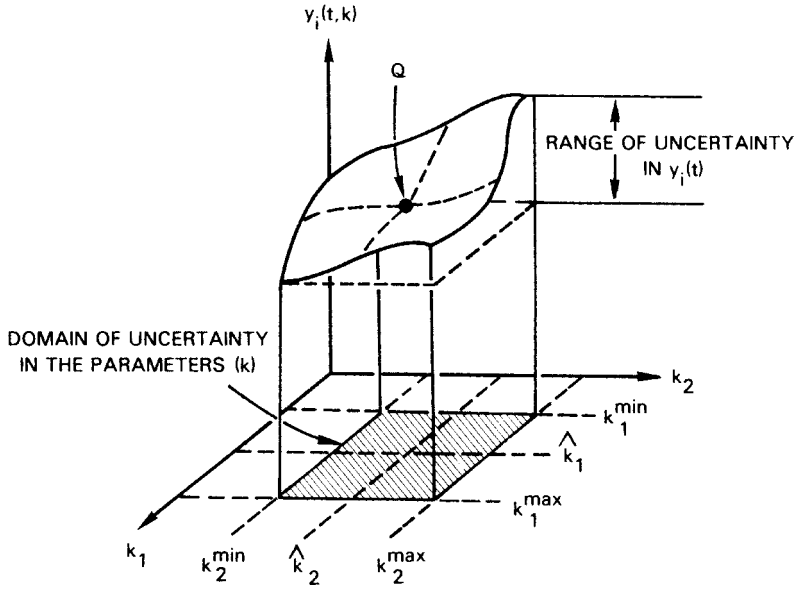

Figure 5.5. Schematic of the solution surface, $y_i(t, \mathbf{k})$, showing the region of uncertainty in two parameters, k_1 and k_2 (Gelinas and Vajk 1978).

When the variation of y_i with k_j is being considered, all other rates $k_l, l \neq j$, are held fixed. Local sensitivity analysis focuses on the calculation and interpretation of these sensitivity coefficients.

Figure 5.5, a schematic diagram of a solution surface $y_i(t, \mathbf{k})$ based on Gelinas and Vajk (1978), shows the region of uncertainty in two parameters, k_1 and k_2. The estimated parameter values are $\widehat{k}_1$ and $\widehat{k}_2$. The assumed upper and lower limits of variation produce a region of uncertainty in the $k_1 - k_2$ plane, which correspond to a range of uncertainty in y_i. The sensitivity coefficients, β_{ij}, evaluated at $\widehat{k}_1$ and $\widehat{k}_2$, represent the slopes of the surface in the two coordinate directions at point Q.

There are several approaches to calculating first-order sensitivity coefficients. The simplest is a "brute-force" finite-difference approach. This involves varying each parameter, one at a time, to compute the deviation of y_i arising from a small change in k_j at the beginning of and throughout the integration. Then

$$\Delta y_i(t, \widehat{\mathbf{k}}) = y_i(t, \widehat{k}_1, \ldots, \widehat{k}_j + \Delta k_j, \ldots, \widehat{k}_m) - y_i(t, \widehat{k}_1, \ldots, \widehat{k}_j, \ldots, \widehat{k}_m). \quad (5\text{–}4.6)$$

The y_i are obtained by integrating equation (5–4.3). This way of determining $\{\Delta y_i(t, \mathbf{k})\}$ depends on the particular problem chosen for integration, and thus cannot lead to any generalizations about sensitivity. This approach can be used for arbitrary variations in Δk_j and is not limited to small variations in the uncertainties. It leads to the approximate first-order sensitivity coefficients,

$$\beta_{ij}^{(o)} \equiv \frac{\Delta y_i(t, \widehat{\mathbf{k}})}{\Delta k_j}. \quad (5\text{–}4.7)$$

This is the most direct approach to finding the effect of varying a parameter, and is certainly

the simplest and most often used method. This method allows us to construct a solution surface in parameter space with the parameters varied systematically (see Box, Hunter, and Hunter [1978]). This approach becomes cumbersome and expensive as the number of species and parameters increase.

Alternative local approaches involve "direct solution," where the sensitivity coefficients themselves are considered as dynamic variables. Differential equations for first-order sensitivity coefficient vectors can be derived using

$$\frac{d\boldsymbol{\beta}_j}{dt} = \mathbf{J} \cdot \boldsymbol{\beta}_j(t) + \mathbf{b}_j(t), \qquad \boldsymbol{\beta}_j(0) = 0. \tag{5–4.8}$$

Equation (5–4.8) is a set of M, N-dimensional vector ODEs where $\mathbf{J}$ is the $N \times N$ Jacobian,

$$J_{il} \equiv \frac{\partial f_i}{\partial y_l}. \tag{5–4.9}$$

The quantities $\{\boldsymbol{\beta}_j\}$ and $\{\mathbf{b}_j\}$ are N-dimensional vectors, such that $\boldsymbol{\beta}_j = (\beta_{1j}, \ldots, \beta_{Nj})$ and $\mathbf{b}_j = (\partial f_i/\partial k_j, \ldots, \partial f_N/\partial k_j)$. We could solve equation (5–4.8) directly in conjunction with equation (5–4.3). This amounts to solving $2N$ equations M times. Otherwise, we can solve equation (5–4.3) to produce $\mathbf{y}(t, (k))$ and then interpolate values of $\mathbf{y}(t, (k))$ to use in $\mathbf{J}$. These approaches are discussed in Tilden et al. (1981). Solving these equations provides systematic information on the importance of small variations about $\mathbf{k}$ for a particular problem, but the results are not necessarily valid for large uncertainties or for different problems.

Another local method that is now widely used is a Green's function method (Kramer, Kee, and Rabitz 1984a; Kramer et al. 1984b). This approach is more efficient than the direct solutions of equation (5–4.8) when there are more parameters whose sensitivity is in question than there are dependent variables. The solution to equation (5–4.8) can be written as

$$\boldsymbol{\beta}_j(t) = \boldsymbol{\psi}(t, 0)\boldsymbol{\beta}_j(0) + \int_o^t \boldsymbol{\psi}(t, \tau)\mathbf{b}_j(\tau)\, d\tau, \tag{5–4.10}$$

where $\boldsymbol{\psi}$ is determined by the equation

$$\frac{d\boldsymbol{\psi}(t, \tau)}{d\tau} + \boldsymbol{\psi}(t, \tau)\mathbf{J}(\tau) = 0, \qquad \boldsymbol{\psi}(t, t) = (\mathbf{I}). \tag{5–4.11}$$

Efficient computational methods involving recursion relations in time have been developed for using this method (Rabitz et al. 1983; Kramer et al. 1984a,b).

Global Methods

Consider Figure 5.5 again. For small displacements about the estimated parameter values, the tangent plane at Q differs by only a small amount from the actual solution surface. The sensitivity coefficients at Q do not contain information on the behavior of the surface away from Q, nor do they indicate the full range of variation of y_i in the region of uncertainty of the parameters.

Global methods calculate averaged sensitivities that quantify the effects of simultaneous, large variations in $\mathbf{k}$. These sensitivities are a qualitatively different measure from the

local sensitivity coefficients discussed above. They are based on the assumption that the uncertainty in k_j can be expressed in terms of a probability density function, p_{y_i} for the dependent variable y_i. These can be used to evaluate statistical properties of the system, such as the expected value,

$$<y_i(t)> = \int_{k_M^{\min}}^{k_M^{\max}} \cdots \int_{k_1^{\min}}^{k_1^{\max}} p_{y_i}(t, \mathbf{k}) p_1(k_1) p_2(k_2) \cdots dk_1 dk_1 \cdots dk_M \qquad (5\text{–}4.12)$$

where $\{p_i(k_i)\}$ are the probability distribution functions of the $\{k_i\}$. The sensitivity analysis then involves computing the functions $\mathbf{p}$. Suppose we have some knowledge of the probability distributions of two parameters, p_{k_1} and p_{k_2}. Then the probability distribution of y_i can be calculated. Whether or not the probability distributions for k_1 and k_2 are given, the solution surface for y_i can be determined by systematically selecting test points in the $k_1 - k_2$ plane and solving the system to determine $y(t; k_1, k_2)$. A global sensitivity analysis requires sampling over the range of uncertainty of the parameters.

Costanza and Seinfeld (1981) developed the *stochastic approach* based on the probability distribution function. First rewrite equation (5–4.3) as

$$\frac{d\mathbf{x}}{dt} = \mathbf{F}(\mathbf{x}, t) \quad \text{and} \quad \mathbf{x}(0) = \mathbf{x}_o, \qquad (5\text{–}4.13)$$

where $\mathbf{x}$ is a vector whose $N + M$ components are the N species concentrations $\mathbf{y}$ and the M parameters $\mathbf{k}$. Then $F_i(\mathbf{x}, t) = f_i(\mathbf{y}, t)$ for $i = 1, \ldots, N$ and $F_{i+N} = 0$ for $i = 1, \ldots, M$. If we assume that the uncertainties in $\mathbf{k}$, including those in initial conditions, can be represented by a probability distribution, then the initial conditions $\mathbf{x}_o$ are random variables with the probability distribution $p_o(\mathbf{x}_o)$ for $i = N + 1, \ldots, N + N$.

Let $p(\mathbf{x}, t)$ be the probability distribution of $\mathbf{x}(t)$. Then

$$<g(\mathbf{x}(t))> = \int d\mathbf{x}\, p(\mathbf{x}, t) g(\mathbf{x}). \qquad (5\text{–}4.14)$$

Costanza and Seinfeld (1981) show that $p(\mathbf{x}, t)$ satisfies

$$\frac{\partial p(\mathbf{y}, t)}{\partial t} + \nabla_{\mathbf{y}} \cdot (p(\mathbf{y}, t)\, \mathbf{f}(\mathbf{y}, t)) = 0, \qquad p(\mathbf{y}, 0) = p_o(\mathbf{y}). \qquad (5\text{–}4.15)$$

Thus, given $p_o(\mathbf{y}) = p_o(\mathbf{x}_o, \mathbf{k})$ and the set of ODEs, equation (5–4.15) can be solved for $p(\mathbf{y}, t)$. Once we know p(**x**, t), this surface in **x**-space can be studied for fixed t. This is, however, difficult for large $N + M$, and so it is useful to study reduced probability distributions obtained by integrating p over a subset of the components of $\mathbf{x}$.

Sometimes there is not enough information about the initial state or the parameters to define p_o, but there are the estimated values of the parameters and an estimate of their range of uncertainty. Then each parameter can be sampled over its range to assess the range of variation of $\mathbf{y}(\mathbf{t})$. This leads to the pattern search procedures: Monte Carlo methods (see, for example, Stolarski, Butler, and Rundel [1978]); pattern methods (see, for example, Sobol [1979]); and Fourier methods (see, for example, Cukier et al. [1973], McRae, Tilden, and Seinfeld [1982]). These methods choose the specific sampling point differently. In general, Fourier methods require considerably fewer sampling points than Monte Carlo methods, but there are more questions about the interpretation of Fourier sensitivity analyses.

General Thoughts on Sensitivity Analysis

Sensitivity methods can provide relationships among the variables and nominal constants in the system. If a set of equations is presented to someone who has little feeling for the important parameters controlling that system, sensitivity analysis is one way of helping to understand the system. When the system being studied is fairly well understood, directly varying selected parameters and recalculating might be the best, and is certainly the simplest way to proceed.

Although knowing the probability distribution functions completely allows the calculation of everything about the system, this would be extremely expensive to implement in practice. Therefore most practical methods try to extract more limited pieces of information, obtaining computational efficiency by trading off other things. It is important to realize exactly what the benefits of each sensitivity analysis methodology are before choosing one.

The most popular methods are the brute-force method and the Green's function approach. The brute-force methods are often combined with clever ways of extracting information. An example of this approach is the direct decoupled method summarized by Radhakrishnan (1991). The Green's function method (Rabitz et al. 1983; Kramer et al. 1984a,b) is also commonly used and is now incorporated as part of the CHEMKIN package described in Section 5–4.3.

5–4.3. Integration Packages for Solving ODEs

In general, there are two classes of methods for solving coupled ODEs, some of which are stiff:

1. methods that treat all of the differential equations, both stiff and nonstiff, identically, and
2. *hybrid methods* that attempt to identify the stiff equations or the key terms in them and treat them with one method, while treating the rest of the equations with a faster but less stable, classical algorithm.

Hybrid methods try to take advantage of the structure of the particular set of ODEs and thus are problem dependent. They are precursors of the quasisteady methods discussed in Section 5–3.3.

Because two methods are being combined, strict conservation of atoms is difficult to guarantee. In hybrid methods, a criterion must be established for determining which method to use for which equation (or for which terms). This requires determining which equations are stiff at each step. A closely related problem is determining when, during the integration, the method should be changed. Changing the timestep can shift some equations from being normal to being stiff. We also need to consider how much nonconservation can be tolerated. Hybrid methods are fast, but because some conservation errors may occur, they often must be independently renormalized.

Some of the currently used software packages for solving coupled sets of ODEs, some or all of which are stiff, are now discussed. We have tried to differentiate between those packages that are best for stand-alone integration of ODEs, packages best for including in reactive-flow calculations, and other useful packages that are generally available, such as those that perform sensitivity analyses. Most of these packages are being updated, even

as this book is being written. Sometimes the older version may be easier to use, as it tries to serve fewer purposes and there is less to learn to get started.

Programs Based on the Gear Method

Subroutines originally written by Gear (1969, 1971) solve sets of coupled ODEs, some of which are stiff, by treating the entire set of equations as either stiff or nonstiff. Stiff equations are solved by the variable-order backward differentiation formulas described in Section 5–3. Normal equations are solved by a variable-order Adams predictor-corrector method described in Section 5–2. These routines have been rewritten to create various packages for large, stiff problems (Hindmarsh 1974, 1976a,b, 1977). The most easily obtained version of the Gear programs has been incorporated in the package ODEPACK (Hindmarsh 1983; Cohen and Hindmarsh 1996). These packages use an implicit Adams-Moulton formula to advance the nonstiff equations and an implicit backward differentiation method to solve the stiff equations. ODEPACK contains a number of other ODE solvers as well. For example, the solver LSODE works for stiff and regular equations. In addition to solving the equations with a full Jacobian, it also allows solutions with various approximations to the Jacobian and is more flexible than the original Gear solvers.

Many general packages now incorporate versions of the Gear method, and in particular, LSODE. These include, for example, CHEMKIN and LENS, which are described below. LSODE can be extremely accurate and you can usually rely on its giving a good answer. It can also be very expensive and is therefore impractical for a reactive-flow computation. For a reactive-flow problem, LSODE may provide the accurate answer against which to test the lower-order, self-starting method that should be used.

An Asymptotic Integration Method: CHEMEQ and VSAIM

The selected asymptotic integration method (SAIM) was designed specifically for use in reactive-flow simulations where the chemical reactions must be described for each spatial cell and must be restarted after each fluid dynamic timestep. In these cases the storage and time costs are major factors. SAIM has also been used extensively for "single-point" problems where a standard stiff ODE package might be expected to work well. The SAIM is implemented in a hybrid integrator, CHEMEQ (Young and Boris 1977; Young 1979), which solves regular equations by an explicit Euler method and uses the asymptotic method discussed in Section 5–3 for the stiff equations. VSAIM is a version of the same program designed specifically for reactive-flow models, because it vectorizes the solution procedure over a number of computational cells simultaneously. A chemical rate processor is included that reads chemical equations in standard symbolic notation and sets up a Fortran program to evaluate the derivatives automatically. Software interfaces, such as this chemical rate processor, are valuable and even necessary additions to all reactive-flow modeling efforts.

A crucial part of CHEMEQ and VSAIM determines criteria for the initial stepsize and identifies the stiff equations. The initial stepsize is

$$\Delta t = \epsilon \min \left\{ \left. \frac{n_i}{Q_i - \mathcal{L}_i n_i} \right|_{t=0} \right\}. \tag{5–4.16}$$

Here ϵ is a scale factor, typically the same as the convergence criterion, and i indexes the initial values of each component of the vector $\mathbf{y}$. If this stepsize is greater than a specified

value τ_i, the equation is considered stiff and is integrated according to the asymptotic formula. Equations considered stiff at the beginning are treated as stiff throughout the current integration step. A fixed, small number of iterations are done each step, and whether or not convergence is achieved determines the next stepsize. It is best to reduce the stepsize sharply (a factor of two or three) when the equations do not converge, and to increase the stepsize gradually, say by 5 to 10 percent, when convergence is achieved. During several successive integration steps, the appropriately modified stepsize from the converged integration cycle is used as the trial stepsize for the next integration cycle rather than reapplying the starting formula, equation (5–4.16).

The method is generally efficient, accurate, and stable. It is self-starting, requiring data only from the previous timestep to initiate the current timestep. It does not use the Jacobian, and therefore does not require matrix operations or inversions. As described in Chapter 4, an asymptotic method may actually be more accurate for large stepsizes. It is not inherently conservative. Conservation is controlled by adjusting the convergence criterion and the criterion for choosing which equations to treat as stiff. Thus the stepsize must be carefully monitored to insure accuracy, convergence, and adequate conservation. The greatest problem with this method is integrating very stiff systems as they approach equilibrium. In this regime, the production and loss terms cancel almost perfectly. In the CHEMEQ method, this situation can lead to large fluctuations in the estimated derivatives. In computing reactive flows, this near equilibrium condition generally causes difficulties. Fluid dynamics and other nonlocal effects and source terms associated with the flow will temporarily shift the chemistry much further from equilibrium than can occur in nature because of the presence of the very stiff terms.

CHEMEQ2: QS and Partial Equilibrium

This is a new ODE integration package, available in 2000, based on the quasi-steady-state method described earlier (Mott 1999; Mott et al. 2000, n.d.). The simplest version is not a hybrid approach, as the QSS method works for both stiff and nonstiff equations. Nonetheless, even CHEMEQ2 fails when the the equations are very, very stiff. (This occurs, for example, when thermonuclear reaction rates with temperature dependences of T^{10} are integrated.) A new approach takes advantage of the fact that partial equilibrium among different groups of chemical species occurs through the evolution of the reaction process and uses this information to remove stiff equations from the ODE system (Mott 1999). Adding the partial equilibrium solver puts CHEMEQ2 into the category of a hybrid approach.

The YAS Method

YASS (*Y*et *A*nother *S*tiff *S*olver) is a package based on the low-order single-step extrapolation method described in Section 5–4 (Khokhlov n.d.). This package uses a single method for solving all of the coupled ODEs, making no distinction between those that are stiff and those that are nonstiff. In general, YASS is more expensive to use in a reactive-flow code than CHEMEQ, because it requires the evaluation of the Jacobian. Unlike many methods, however, the Jacobian only has to be evaluated once at each timestep and grid location. In addition, YASS requires no subcycling or iterations within the global timestep. Thus, for example, the fluid timestep can be the timestep used in YASS.

Because YASS is low order and self-starting, it is a good method to use in reactive-flow codes, and a viable alternative when faster methods, such as CHEMEQ, do not work well. This might be the case, for example, when the equations are extremely stiff and good solutions from CHEMEQ would require a large number of subcycles. Generally, however, methods that involve evaluating the Jacobian become inordinately expensive in a reactive-flow code for even a moderately complex reaction mechanism.

Other Useful Software

There are several extremely useful collections of software that are reliable and have become standards for the community. They are generally distributed at minimal or no cost. Used properly, these packages provide important input data and useful checks on solutions developed for a multidimensional reactive-flow problem. For example, the packages might provide thermodynamic data required for the equations of state. They might also be used to check the formally less-accurate solutions obtained by the integration method used in a full reactive-flow code.

The NASA Gordon-McBride program, *Chemical Equilibrium Applications* or CEA (McBride and Gordon 1996), is the latest version of a package that has been a standard in the chemistry, combustion, and propulsion communities for many years. It is used as a source of thermodynamic data and to compute system properties such as the equilibrium chemical composition of a mixture, properties of Chapman-Jouguet detonations and reflected shocks, adiabatic flame temperatures, and so on. Versions exist for different computers ranging from mainframes to workstations to small desktop personal computers. The usefulness of this package for reactive flows cannot be overstated.

CHEMKIN (Kee and Miller 1996; Kee, Rupley, and Miller 1996) refers to a collection of software for integrating and analyzing the properties of large chemical reaction systems. It is primarily for gas-phase chemical kinetics, with capabilities for heterogeneous chemical kinetics. The CHEMKIN software consists of packages to:

- interpret the chemical reactions and rates when they are supplied in a specific format by the user;
- use an internal data base to find the relevant thermodynamic data; and
- integrate the resulting ODEs, based on LSODE.

Additional packages, such as SENKIN (Lutz, Kee, and Miller 1987), perform sensitivity analyses or solve a selection of steady-state combustion problems. This collection of programs is representative of a structured modeling philosophy that has proved extremely useful in combustion research, and it is an excellent example of the type of modular programming described in Chapter 3. The CHEMKIN representation of sets of chemical reactions has become the standard way that chemical reaction mechanisms are transmitted and codified.

LENS (Radhakrishnan 1994) was developed for homogeneous, gas-phase chemical kinetics computations. It contains sensitivity analyses for a variety of problems. As a general software tool, it may be used for the same types of problems as CHEMKIN. LENS can be used to model a static system, a steady, one-dimensional inviscid flow, reaction behind an incident shock wave, or a perfectly stirred reactor. In addition, LENS can provide

a sensitivity analysis for static situations. The ODE integration is based on LSODE, and the sensitivity analysis is based on a decoupled direct method (see Radhakrishnan [1991]) that solves the sensitivity equations separately from, but sequentially with, the model equations.

5–4.4. Reaction Kinetics and Thermodynamic Input Data

Thermochemical data refers to the heats of formation and the specific heats (c_p and c_v) of various materials. These essential parts of reactive-flow codes are input for the equation of state, an essential part of reactive-flow calculations. Except for perfect gases, the specific heats are functions of temperature, and so are usually fit to polynomials that may go up to fourth or fifth order. The standard sources of this data are the JANAF and CEA tables, and CHEMKIN. These tables are not all independent. For example, CHEMKIN draws from CEA, which has been accumulating data for some time. For applications in reactive-flow codes, it is important to have smooth data over the entire temperature range. Thus expressions that involve several polynomial fits to cover a temperature range usually have to be refit to a single (perhaps higher-order) polynominal.

Where to find input for a chemical reaction mechanism is a massive subject. Different fields have their own particular sources. For hydrocarbon combustion, there are standard references, such as the GRI (Gas Research Institute) or NIST (formerly, National Bureau of Standards [NBS]) tabulations, which are often a good place to start. JPL (Jet Propulsion Laboratory) has compiled data for atmospheric chemistry. Thermonuclear-reaction mechanisms can be found from scanning the *Astrophysical Journal*. These mechanisms are often best found on the Internet using a Web browser. In general, finding a reaction mechanism means searching the literature. It is useful, however, to consult colleagues who might provide good references and some indication of the limitations of a mechanism before it is used.

5–5. Reduced Reaction Mechanisms for Reactive Flows

As Table 5.6 shows, the size of a detailed reaction mechanism can become so large that integrating the representative ODEs directly in a time-dependent, multidimensional reactive-flow code becomes ridiculously expensive, even with fast, special purpose ODE integrators. Substantial efforts to deal with this problem have yielded different approaches that fall into a set of overlapping categories:

- methods that fit data or parameters from detailed mechanisms into a given form or a table;
- methods that use analytical techniques and knowledge of the chemical reaction mechanism to derive simplified sets of reaction-like equations for a particular mechanism; and
- methods that eliminate the stiffness problem in the ODEs, sometimes by using partial equilibrium or steady-state concepts.

A recent summary of many of these methods has been given in the review articles by Griffiths (1995), Peters and Rogg (1993), and the collection of articles edited by Smooke (1991). This section now summarizes features of several of these methods that have been found useful in different types of reactive-flow simulations.

5–5.1. The Induction Parameter Model

The induction parameter model (IPM) is a useful chemistry phenomenology for numerical simulations (Oran et al. 1980; Oran and Boris 1982; Oran, Boris, and Kailasanath 1991; Lefebvre, Oran, and Kailasanath 1992). This method proposes a global form for the reaction, assumed to be occuring over a finite time, and then fits parameters using data from experiments, detailed numerical simulations, or a combination. The input data then appear either in the form of a table or an analytic expression. The basic concept of an IPM is that as a fluid element moves, it experiences changing temperatures and pressures that alter the conditions under which the chemical reaction proceeds. In its most common form, the IPM has proved very useful in simulations of detonations and other high-speed reactive flows, for which physical diffusion effects are relatively slow compared to the chemical reaction steps. Generalizations of these concepts have been considered to include effects of diffusion and multispecies chemical behavior.

For example, consider a two-step model for ignition of a combustible mixture. The first step represents a chemical induction period in which fuel and oxidizer interact to produce a substantial amount of chemical intermediates (Step 1). Then Step 2 models the reaction of the fuel, oxidizer, and intermediates (perhaps chemical radicals) to form products. The first step might be thermoneutral, endothermic, or even mildly exothermic, depending on the system represented. In combustion, the second step is usually the exothermic step leading to a final state that depends on the initial state of the material and on states through which the material has passed during the reaction period. When the concentration of intermediates is low, they are produced by Step 1 much faster than they are consumed by Step 2. This is a period in which the number of intermediates increases rapidly but their number is still relatively small, so that the amount of energy released is small.

The rate of Step 1 can be expressed in terms of an induction time $\tau_o(T, \rho)$. One way of finding $\tau_o(T, \rho)$ is to integrate a full set of detailed chemical equations; another is to extract it from experimental data. The induction period is complete when appreciable products from Step 2 begin to appear. This is the approach used, for example, in detonation calculations for hydrogen-oxygen systems (see, for example, Oran et al. [1980, 1981], Kailasanath et al. [1985], Lefebvre et al. [1993], and most recently, Tonello, Sichel, and Oran [n.d.]). Another approach that is useful when the detailed chemical mechanism is not known is to use experimental data (for example, Guirguis, Oran, and Kailasanath [1986, 1987]). In some applications of this generic approach, the induction time includes considerably more than a chemical reaction period. It has been used in studies of the properties of droplets and sprays in flows (for example, Bar-Or, Sichel, and Nicholls [1982]). In this case, the computational induction time is more complex and also involves a droplet or spray evaporation time. In recent studies, a generalized IPM is used to include effects of large-scale mixing.

Let f denote the fraction of the induction time elapsed at time t. Then

$$\frac{df}{dt} = \frac{1}{\tau_o(T, \rho)}, \tag{5–5.1}$$

where $f(\mathbf{x}, 0) = 0$. The rate of Step 2 is kept at zero until f exceeds unity. This formula provides a convenient way of averaging over changing temperature and density during the chemical induction period. Equation (5–5.1) reduces automatically to the correct temperature-dependent induction delay when the density and temperature are constant. Once the induction delay has elapsed, the fuel and oxidizer are consumed according to

$$\frac{d}{dt} N_{\text{fuel}} = -N_{\text{fuel}}\, A \exp(-E/RT) \tag{5–5.2}$$

where N_{fuel} is the density of fuel. In this model, f can be interpreted as the fraction of the critical radical concentration. When this model is used in a moving system, d/dt in equations (5–5.1) and (5–5.2) denotes the derivative following the fluid flow. The quantity f must be advected with the fluid but is not compressed.

Some of the basic ideas of the IPM have been incorporated in more mathematically sophisticated approaches. For example, chemical reaction mechanisms are sometimes reduced based on an analysis of their time scales, so that the faster and slower processes are separated from each other. Sometimes reduction methods decide in advance which information needs to be included to give the important effects in a reactive-flow calculation. Collecting precomputed material into tables to represent chemical data is becoming more common in reactive-flow calculations. Some of the other, more complex approaches are summarized in the following sections.

5–5.2. Computational Singular Perturbations

Computational singular perturbations (CSP) is a method that begins with a detailed reaction mechanism and, using partial equilibrium concepts, attempts to eliminate the stiffness problem altogether. This works because some of the reactions are so stiff that they can be assumed to be in an evolving quasi–steady state that closely tracks the slow changes in the nonstiff components of the chemical reactions.

CSP originally began as an attempt to automate singular perturbation analysis (see, for example, O'Malley [1991]) for stiff systems of ODEs (Lam 1985). As such, it was originally used to study complex chemical mechanisms (Lam and Goussis 1988, 1992; Lam 1993). This work constructed reduced chemical mechanisms and identified approximate equations of state reflecting partial equilibrium as it occurred in the system. The CSP analysis also identified the dominant reactants and reactions in the mechanism, and so provided sensitivity analyses. Current implementations of CSP use this information to reduce the number of ODEs that must be integrated by neglecting relatively unimportant species and chemical source terms. The result is that the stiff ODE system is replaced with a reduced, nonstiff ODE system (Lam and Goussis 1994; Lam 1995). To date, CSP has evolved into a potentially useful method for integrating stiff systems of equations. The methods are now being extended to reactive-flow problems (Valorani and Goussis n.d.; Hadjinicolaou and Goussis 1998).

The starting point for the CSP analysis is a complete set of chemical reactions and chemical reaction rates. Given N species, equation (5–1.2) represents the system of N ODEs. The source term, f, is a vector of dimension N and is usually written as the sum over all reactions in the chemical mechanism

$$f = \sum_{r=1}^{R} s_r F^r, \tag{5–5.3}$$

where F^r and s_r are the net reaction rate and the stoichiometric vector, respectively, of the r^{th} reaction, and R is the total number of reactions in the mechanism. Since the N-dimensional space representing all possible species concentrations can be spanned by any set of N linearly independent basis vectors, f can also be written as

$$f = \sum_{n=1}^{N} a_n G^n, \tag{5–5.4}$$

where each a_n is an N-dimensional vector. Each of these vectors represents a reaction "mode," and the weight of each vector, G^n, is a linear combination of the $\{F^r\}$. For equation (5–5.4) to be exact, the set $\{a_n\}$ must be a complete set that includes all of the relevant interacting species.

CSP splits the modes into groups, for example,

$$f = \sum_{n=1}^{M} a_n G^n + \sum_{n=M+1}^{N} a_n G^n = f_{\text{fast}} + f_{\text{slow}}. \tag{5–5.5}$$

The first group ($1 \le n \le M$) involves the fast time scales, and the amplitudes of these modes (that is, their contribution to the source term) fall exponentially as time progresses. These modes are responsible for the fast, transient behavior of the system. The second group ($M + 1 \le n \le N$) involves the slow modes and is the group responsible for the slow evolution of the species in the vicinity of the quasi-steady-state solution determined by the relaxation of the fast modes. The fast modes can be thought of as quickly driving the solution to an $(N - M)$-dimensional manifold in composition space on which the solution slowly evolves; this evolution is governed by the slow modes.

Once the fast modes have decayed sufficiently (i.e., once the composition has reached the slow manifold), they can be neglected in the ODE integration. Neglecting f_{fast} removes the stiffness of the system, so we may solve

$$\frac{dy}{dt} \approx f_{\text{slow}}, \tag{5–5.6}$$

and also gives the algebraic relationships of equilibrium conditions among the species,

$$G^n \approx 0, \qquad 1 \le n \le M. \tag{5–5.7}$$

For f_{fast} to be neglected in the integration, the contribution of the fast modes to y must fall below a prespecified tolerance vector, element by element. Such criteria allow the user to specify varying accuracy requirements on different species.

For these simplifications to work, the set of basis vectors a_n must be chosen carefully. CSP automatically finds the fast and slow groupings by refining a given basis set to better decouple the fast modes from the slow modes (Lam and Goussis 1994). This updating procedure also allows the a_n's and the value of M to evolve as the dynamics of the system change. If the fast and slow modes could be completely decoupled, the solution would stay on the slow manifold after being placed there by the fast modes. Because of the nonlinearity in the ODE system, the fast-mode manifold is continually changing as the solution evolves, so the definitions of the slow modes must be changed accordingly to prevent small components of the fast modes from creeping back into the solution and destabilizing the slow-mode integration. Either of two approaches can be employed to prevent the accumulation of this error. In the first, after integrating the set of N ODEs (using equation (5–5.6)), a "radical correction" is used to drive the composition back toward the manifold by satisfying the algebraic equilibrium constraints (equation (5–5.7)) to first order. The second method starts by using the CSP-derived data to identify the M species most accurately calculated by the algebraic equilibrium relations. The other $N - M$ species are found through integration, and then the remaining M species are found by solving the coupled set of M nonlinear algebraic equations given by equation (5–5.7).

CSP provides valuable insights into the dynamics of a stiff ODE system and uses this knowledge to integrate the system. This information, however, is not free. There is considerable overhead in refining the basis vectors as the solution evolves, and in storing this information. The detailed chemical mechanism is replaced in the ODE integration by the CSP-derived system, but the detailed system is still needed for the basis refinement. The "danger" of giving up dynamic refinement is losing reaction pathways that might become important later in the evolution of the system. The most straightforward application of CSP to a reactive-flow problem requires that these basis vectors be refined and stored for every cell in the computational domain. As such, the chemical system under study must be sufficiently stiff or else CSP does not provide a pay-off relative to using more standard stiff integrators.

5–5.3. Intrinsic Low-Dimensional Manifolds

Intrinsic low-dimensional manifolds (ILDM) uses the full chemical reaction mechanism as a basis for a formal reduction process (Maas and Pope 1992a,b, 1994). In fact, the starting point for ILDM and CSP is essentially the same, equations (5–5.3) through (5–5.5). The objective of ILDM is a shortened chemical reaction mechanism, with enough fewer species and reactions that it is suitable for use in reactive-flow computations. Ultimately, if the mechanism is reduced enough, it could be used to construct tables that might replace ODE integrations. Thus the approach is in some ways very similar to CSP, but the objective and limitations are more along the lines of the IPM.

The description of the IPM, CSP and ILDM approaches emphasized that there are times when the long-time behavior of the variables in sets of ODEs can often be described by a lower-dimensional (fewer eigenvectors, fewer modes) set of equations. This lower-dimensional manifold may be identified with the longer time scales of the kinetics, once the fastest modes have died away. Then the low-dimensional manifold is sought by assuming partial equilibrium of certain reactions, or by analyzing the system when it reaches a

stationary state with respect to the reactive species. Different initial assumptions, however, produce different manifolds. But once the initial constraints desired from the solution are specified, the reduction process can proceed by a formal route.

Because the objectives for the use of CSP and ILDM are somewhat different, the methods tend to be more complementary than conflicting. ILDM will produce a reaction mechanism of prespecified size that can be used in a reactive-flow code. This mechanism or the table derived from it has inherent inaccuracies that may or may not become important, depending on what the problem is and how big a mechanism is retained. CSP does not predetermine the size of the manifold: it lets the evolution of the system determine this self-consistently. As such it provides automatic changes in dimension if the system undergoes an unforeseen change. CSP provides a built-in sensitivity analysis to determine this as well as the importance of other modes and species. The cost of using CSP can be large, but it can also be used to derive a reduced mechanism and so overlaps with ILDM in intent. Because of the simplicity of the form of the result, ILDM has been used directly coupled in a number of different types of reactive-flow simulations (see, for example, Maas and Pope [1994]; Maas [1995]; Schmidt et al. [1996]).

REFERENCES

Arnett, W.D., and J.W. Truran. 1969. Carbon-burning nucleosynthesis at constant temperature. *Astrophysical Journal* 157:339–365.

Babcock, P.D., L.F. Stutzman, and D.M. Brandon, Jr. 1979. Improvements in a single-step integration algorithm. *Simulation* 33:1–10.

Bar-Or, R., M. Sichel, and J.A. Nicholls. 1982. The reaction zone structure of cylindrical detonations in monodisperse sprays. In *Proceedings of the Nineteenth Symposium (International) on Combustion*, 665–673. Pittsburgh, Pa.: The Combustion Institute.

Bashforth, F., and J.C. Adams. 1883. *An attempt to test the theories of capillary action by comparing the theoretical and measured forms of drops of fluid. With an explanation of the method of integration employed in constructing the tables which give the theoretical forms of such drops*. Cambridge, England: University Press.

Box, G.E.P, W.G. Hunter, and J.S. Hunter. 1978. *Statistics for experimenters*, New York: Wiley.

Brandon, D.M., Jr. 1974. A new single-step implicit integration algorithm with A-stability and improved accuracy. *Simulation* 23:17–29.

Bui, T.D., A.K. Oppenheim, and D.T. Pratt. 1984. Recent advances in methods for numerical solution of ODE initial value problems. *J. Comput. Appl. Math.* 11:283–296.

Bulirsch, R., and J. Stoer. 1964. Fehlerabschätzungen und extrapolation mit rationalen funktionen bei verfahren vom Richardson-typus. *Numer. Math.* 6:413–427.

———. 1966. Numerical treatment of ordinary differential equations by extrapolation methods. *Numer. Math.* 8:1–13.

Byrne, G.D., and A.C. Hindmarsh. 1987. Stiff ODE Solvers: A review of current and coming attractions. *Journal of Computational Physics* 70:1–62.

Burrage, K. 1995. *Parallel and sequential methods for ordinary differential equations*. Oxford, England: Clarendon Press.

Cohen, S.D., and A.C. Hindmarsh. 1996. CVODE, a stiff/nonstiff ODE solver in C. *Computers in Physics* 10:138–143.

Costanza, V., and J.H. Seinfeld. 1981. Stochastic sensitivity analysis in chemical kinetics, *J. Chem. Phys.* 74:3852–3858.

Cryer, C.W. 1973. A new class of highly-stable methods: A_o-stable methods. *BIT* 13:153–159.

Cukier, R.I., C.M. Fortuin, K.E. Shuler, A.G. Petschek, and J.H. Schaibly. 1973. Study of the sensitivity of coupled reaction systems to uncertainties in rate coefficients. I. Theory. *J. Chem. Phys.* 59:3873–3878.

Curtis. A.R. 1978. Solution of large, stiff initial value problems – The state of the art. In *Numerical software – Needs and availability*, ed. D. Jacobs, 257–278. New York: Academic.

Curtiss, C.F., and J.O. Hirschfelder. 1952. Integration of stiff equations. *Proc. Nat. Acad. Sci.* 38:235–243.

Dahlquist, G.G. 1963. A special stability problem for linear multistep methods. *BIT* 3:27–43.

Dahlquist, G.G., and B. Lindberg. 1973. *On some implicit one-step methods for stiff differential equations*. Technical Report TRITA-NA-7302. Stockholm: Royal Institute of Technology.

Ehle, B.L. 1969. *On pade approximations to the exponential function and A-stable methods for the numerical solution of initial value problems*. Research report no. CSRR 2010. Waterloo, Canada: Department of Applied Analysis and Computer Science, University of Waterloo.

Gear, C.W. 1969. The automatic integration of stiff ordinary differential equations. In *Information Processing 68*, Proceedings of the IFIP Congress. ed. A.J.H. Morrell, 187–193. Amsterdam: North-Holland.

———. 1971. *Numerical initial value problems in ordinary differential equations*, Englewood Cliffs, N.J.: Prentice Hall.

Gelinas, R.J., and J.P. Vajk. 1978. *Systematic sensitivity analysis of air quality simulation models*. PB80-112162. Pleasanton, Calif.: Science Applications Inc.

Gill, S. 1951. A process for the step-by-step integration of differential equations in an automatic digital computing machine. *Proc. Cambridge Philos. Soc.* 47:96–108.

Gordon, S., and B.J. McBride. 1976. *Computer program for calculation of complex chemical equilibrium compositions, rocket performance, incident and reflected shocks, and Chapman-Jouguet detonations*. NASA SP-273. Washington, D.C.: National Aeronautics and Space Administration.

Gragg, W.B. 1965. On extrapolation algorithms for ordinary initial value problems. *SIAM J. Numer. Anal.* 2:384–403.

Griffiths, J.F. 1995. Reduced kinetic models and their applications to practical combustion systems. *Progress in Energy and Combustion Systems* 21:25–107.

Guirguis, R., E.S. Oran and K. Kailasanath. 1986. Numerical simulations of the cellular structure of detonations in liquid nitromethane – Regularity of the cell structure. *Combustion and Flame* 65:339–366.

———. 1987. The effect of energy release on the regularity of detonation cells in liquid nitromethane. In *Proceedings of the 21st Symposium (International) on Combustion*, 1659–1668. Pittsburgh: The Combustion Institute.

Hadjinicolaou, M., and D.A. Goussis. 1998. Asymptotic solution of stiff PDE's with the CSP method – The reaction diffusion equation. *SIAM Journal of Scientific Computing*. 20:781–810.

Hairer, E., S.P. Norsett, and G. Wanner. 1987. *Solving ordinary differential equations I: Nonstiff problems*. New York: Springer-Verlag.

Hairer, E., and G. Wanner. 1991. *Solving ordinary differential equations I: Nonstiff problems*. New York: Springer-Verlag.

Hindmarsh, A.C. 1974. *GEAR: Ordinary differential equation system solver*. LLNL report UCID-30001, rev. 3. Livermore, Calif.: Lawrence Livermore National Laboratories.

———. 1976a. *Preliminary documentation of GEARB: Solution of ODE systems with block-iterative treatment of the Jacobian*. LLNL Report UCID-30149. Livermore, Calif.: Lawrence Livermore National Laboratories [UCID-30130]

———. 1976b. *Preliminary documentation of GEARBI: Solution of implicit systems of ordinary differential equations with banded Jacobian*. Livermore, Calif.: Lawrence Livermore National Laboratories.

———. 1977. *GEARB: Solution of ordinary differential equations having banded Jacobian*. LLNL report UCID-30059, rev. 2. Livermore, Calif.: Lawrence Livermore National Laboratories.

———. 1983. ODEPACK, a systematized collection of ODE solvers. In *Numerical methods for scientific computation* ed. R.S. Stepleman, 55–64. New York: North-Holland.

Hockney, R.W. 1965. A fast direct solution of Poisson's equation using Fourier analysis. *J. Assoc. Comput. Mach.* 12:95–113.

Jay, L.O., A. Sandu, A. Porta, and G.R. Carmichael. 1997. Improved quasi-steady-state approximation methods for atmospheric chemistry integration. *SIAM Journal of Scientific Computing* 18:182–202.

Kailasanath, K., E.S. Oran, J.P. Boris, and T.R. Young. 1985. Determination of detonation cell size and the role of transverse waves in two-dimensional detonations. *Combust. Flame* 61:199–209.

Kee, R.J., and J.A. Miller. 1996. *A structured approach to the computational modeling of chemical kinetics and molecular transport in flowing systems*. Sandia report SAND86-8841. Livermore, Calif.: Sandia National Laboratories.

Kee, R.J., F.M. Rupley, and J.A. Miller. 1996. *Chemkin-II: A Fortran chemical kinetics package for the analysis of gas-phase chemical kinetics*. Sandia report SAND89-8009B · UC–706. Livermore, Calif.: Sandia National Laboratories.

Keneshea, T.J. 1967. *A technique for solving the general reaction-rate equations in the atmosphere*. AFCRL-67-0221. Hanscom Field, Mass.: Air Force Cambridge Research Laboratories. [AD 654010]

Khokhlov, A.M. n.d. *YASS – Yet another stiff solver that works*. NRL memorandum report.

Kramer, M.A., R.J. Kee, and H. Rabitz. 1984a. *CHEMSEN: A computer code for sensitivity analysis of elementary chemical-reaction models*. Sandia report SAND-82-8230. Albuquerque, N.Mex.: Sandia National Laboratories.

Kramer, M.A., H. Rabitz, J.M. Calo, and R.J. Kee. 1984b. Sensitivity analysis in chemical kinetics: Recent developments and computational comparisons. *Int. J. Chem. Kin.* 16:559–578.

Lam, S.H. 1985. Singular perturbation for stiff equations using numerical methods. In *Recent advances in the aerospace sciences*. ed. C. Casci, 3–20. New York and London: Plenum Press.

———. 1993. Using CSP to understand complex chemical kinetics. *Combustion Science and Technology* 89:375–404.

———. 1995. Reduced chemistry modelling and sensitivity analysis. In *Aerothermochemistry for hypersonic technology*. 1994-1995 Lecture Series Programme. Brussels: Von Karman Institute for Fluid Dynamics.

Lam, S.H., and D.A. Goussis. 1988. Understanding complex chemical kinetics with computational singular perturbation. In *Proceedings of the Twenty-Second Symposium (International) on Combustion*, 931–941. Pittsburgh: The Combustion Institute.

———. 1992. Study of homogeneous methanol oxidation kinetics using CSP. In *Proceedings of the 24th Symposium (International) on Combustion*, 113–120. Pittsburgh: The Combustion Institute.

———. 1994. The CSP method for simplifying kinetics. *International Journal on Chemical Kinetics* 26:461–486.

Lambert, J.D. 1973. *Computational methods in ordinary differential equations*. New York: Wiley.

———. 1980. Stiffness. In *Computational techniques for ordinary differential equations*, eds. I. Gladwell and D.K. Sayers, 19–46. New York: Academic.

Lefebvre, M.H., E.S. Oran, and K. Kailasanath. 1992. *Computations of detonation structure: The influence of model input parameters.* NRL memorandum report 6961. Washington, D.C.: Naval Research Laboratory.

Lefebvre, M.H., E.S. Oran, K. Kailasanath, and P.J. Van Tiggelen. 1993. The influence of the heat capacity and diluent on detonation structure. *Combustion and Flame* 95:206–218.

Lindberg, B. 1973. *IMPEX2 – A procedure for solution of systems of stiff differential equations*. Technical report TRITA-NA-7303. Stockholm: Royal Institute of Technology.

Liniger, W., and R.A. Willoughby. 1970. Efficient integration methods for stiff systems of ordinary differential equations, *SIAM J. Num. Anal.* 7:47–66.

Lutz, A.E., R.J. Kee, and J.A. Miller. 1996. *SENKIN: A Fortan program for predicting homogeneous gas phase chemical kinetics with sensitivity analysis*. Sandia report SAND87-8248 UC–401. Albuquerque, N.Mex.: Sandia National Laboratories.

Maas, U. 1995. Coupling of chemical reaction with flow and molecular transport. *Appl. Math.* 3:249–266.

Maas, U., and S.B. Pope. 1992a. Simplifying chemical kinetics: Intrinsic low-dimensional manifolds in composition space. *Combustion and Flame* 88:239–264.

———. 1992b. Implementation of simplified chemical kinetics based on intrinsic low-dimensional manifolds. In *Twenty-Fourth Symposium (International) on Combustion*, 103–112. Pittsburgh: The Combustion Institute.

———. 1994. Laminar flame calculations using simplifed chemical kinetics based on the intrinsic low-dimensional manifolds. In *Twenty-Fifth Symposium (International) on Combustion*, 1349–1356. Pittsburgh: The Combustion Institute.

May, R., and J. Noye. 1984. The numerical solution of ordinary differential equations: Initial value problems. In *Computational techniques for differential equations*, ed. J. Noye, 1–94. New York: North-Holland.

McBride, B.J., and S. Gordon. 1996. *Computer program for calculation of complex chemical equilibrium compositions and applications*. NASA reference publications, 1311–1312. Cleveland, Ohio: NASA-Lewis Research Center.

McCalla, T.R. 1967. *Introduction to numerical methods and FORTRAN programming*. New York: Wiley.

McRae, G.J., J.W. Tilden, and J.H. Seinfeld. 1982. Global sensitivity analysis – A computational implementation of the Fourier amplitude sensitivity test (FAST). *Comp. and Chem. Eng.* 6:15–25.

Mott, D.R. 1999. New quasi-steady-state and partial-equilibrium methods for integration chemically reacting systems. Ph.D. diss. The University of Michigan, Department of Aerospace Engineering.

Mott, D.R, E.S. Oran, and B. van Leer. n.d. A new quasi-steady-state solver for the stiff ordinary differential equations of reaction kinetics, submitted to *Journal of Computational Physics*.

———. 2000. *CHEMEQ2: A solver for the stiff differential equations of reaction kinetics*. Naval Research Laboratory Memorandum Report. Washington, D.C.: Naval Research Laboratory. [to appear]

Neville, E.H. 1934. Iterative interpolation, *J. Ind. Math. Soc.* 20:87–120.

O'Malley, R.E., Jr. 1991. *Singular perturbation methods for ordinary differential equations*. Vol. 89 of *Applied mathematical sciences*. New York: Springer-Verlag.

Oran, E., and J. Boris. 1982. Weak and strong ignition: II. Sensitivity of the hydrogen-oxygen system. *Combustion and Flame* 48:149–161.

Oran, E.S., J.P. Boris, and K. Kailasanath. 1991. Studies of detonation initiation, propagation, and quenching. In *Numerical Approaches to Combustion Modeling*. eds. E.S. Oran and J.P. Boris, 421–445. Washington, D.C.: AIAA.

Oran, E., J.P. Boris, T.R. Young, M. Flanigan, T. Burks, and M. Picone. 1980. *Simulations of gas phase detonations: Introduction of an induction parameter model*. NRL Memorandum Report 4255. Washington, D.C.: Naval Research Laboratory.

———. 1981. Numerical simulations of detonations in hydrogen-air and methane-air mixtures. In *Proceedings of the 18th International Symposium on Combustion*. 1641–1649, Pittsburgh: The Combustion Institute.

Page, M., E. Oran, D. Miller, R. Wyatt, H. Rabitz, and B. Waite. 1985. A comparison of quantum, classical and semiclassical descriptions of a model, collinear, inelastic collision of two diatomic molecules. *J. Chem. Phys.* 83:5635–5646.

Peters, N., and B. Rogg. 1993. *Reduced kinetic mechanisms for applications in combustion systems*. Berlin: Springer-Verlag.

Pratt, D.T., and K. Radhakrishnan. 1984. *CREKID: A computer code for transient, gas-phase combustion kinetics*. NASA technical memorandum 83806. Washington, D.C.: National Aeronautics and Space Administration.

Rabitz, H., M. Kramer, and D. Dacol. 1983. Sensitivity analysis in chemical kinetics. *Ann. Rev. Phys. Chem.* 34:419–461.

Radhakrishnan, K. 1984. A comparison of the efficiency of numerical methods for integrating chemical kinetic rate equations. In *Proceedings of the JANNAF Propulsion Meeting*, New Orleans; also NASA technical memorandum 83590. Washington, D.C.: National Aeronautics and Space Administration.

———. 1991. Combustion kinetics and sensitivity analysis computations. In *Numerical approaches to combustion modeling*, ed. E.S. Oran and J.P. Boris. Vol. 135 of *Progress in astronautics and aeronautics*, Washington, D.C.: AIAA.

———. 1994. *LENS – General chemical kinetics and sensitivity analysis code for homogeneous gas-phase reactions*. NASA reference publications, 1328–1330. Cleveland, Ohio: NASA-Lewis Research Center.

Ralston, A. 1978. *A first course in numerical analysis*. New York: McGraw-Hill.

Schmidt, D., J. Setgatz, U. Riedel, J. Warnatz, and U. Maas. 1996. Simulation of laminar methane-air flames using automatically simplified chemical kinetics. *Comb. Sci. Tech* 113–114:3–16.

Shampine, L.F., H.A. Watts, and S.M. Davenport. 1976. Solving nonstiff ordinary differential equations – The state of the art. *SIAM Rev.* 18:376–411.

Sobol, I.M. 1979. On the systematic search in a hypercube. *SIAM J. Numer. Anal.* 16:790–793.

Smooke, M.D. 1991. *Reduced kinetic mechanisms and asymptotic approximation for methane-air flames*. Berlin: Springer-Verlag.

Stolarski, R.S., D.M. Butler, and R.D. Rundel. 1978. Uncertainty propagation in a stratospheric model, 2. Monte Carlo analysis of imprecisions due to reaction rates. *J. Geophys. Res.* 83:3074–3078.

Stull, D.R., and H. Prophet. 1971. *JANAF thermochemical tables*. 2nd ed. National standard reference data series. U.S. National Bureau of Standards, no. 37. Washington, D.C.: Government Printing Office.

Tilden, J.W., V. Costanza, G.J. McRae, and J.H. Seinfeld. 1981. Sensitivity analysis of chemically reacting systems, In *Modelling of Chemical Reaction Systems*, eds. K.H. Ebert, P. Deuflhard, and W. Jäger, 69–91 New York: Springer-Verlag.

Tonello, N., M. Sichel, and E.S. Oran. n.d. Two step kinetics for numerical simulations of explosions and detonations in undiluted H_2-O_2 mixtures, in preparation.

Valorani, M., and D.A. Goussis. n.d. Explicit time-scale splitting algorithms for stiff ODE's: Auto-ignition behind a steady shock. *Journal of Computational Physics*, submitted for publication.

Verwer, J.G., and D. Simpson. 1995. Explicit methods for stiff ODEs from atmospheric chemistry. *Applied Numerical Mathematics* 18:413–430.

Verwer, J.G., and M. van Loon. 1994. An evaluation of explicit pseudo-steady-state approximation schemes for stiff ODE systems from chemical kinetics. *Journal of Computational Physics* 113:347–352.

Wagoner, R.V. 1969. Synthesis of the elements within objects exploding from very high temperatures. *Astrophysical Journal Supplement Series No. 162* 18:247–295.

Weber J.W., Jr., J.D. Anderson, Jr., E.S. Oran, G. Patnaik, and R. Whaley. 1996. Load balancing and performance issues for the data parallel simulation of stiff chemical nonequilibrium flows, *AIAA Journal* 35:4486–4493.

Widlund, O.B. 1967. A note on unconditionally stable linear multistep methods. *BIT* 7:65–70.

Young, T.R. 1979. *CHEMEQ – Subroutine for solving stiff ordinary differential equations*. NRL memorandum report 4091. Washington, D.C.: Naval Research Laboratory.

Young, T.R., and J.P. Boris. 1977. A numerical technique for solving stiff ordinary differential equations associated with the chemical kinetics of reactive-flow problems. *J. Phys. Chem.* 81:2424–2427.

6

Representations, Resolution, and Grids

This chapter returns to the problems of representing a continuous physical variable by a discrete set of numbers, and then using this representation as a basis for solving the equations of the mathematical model. Here the term *computational representation* includes:

- the particular discretization (that is, the *mesh*, *grid*, or *expansion*) used to approximate the continuous flow variables,
- the data structures used to present this discretization to the computer, and
- the interpretation procedure used to reconstruct a numerical approximation of the continuous variable from the set of discrete quantities.

Choosing a computational representation is just as important as choosing a mathematical model to describe the system, or choosing the algorithms to implement that model. For example, the choice of either an Eulerian or Lagrangian representation is important because this choice constrains the type of numerical algorithms and gridding methods that can be used.

This chapter describes the basic concepts underpinning different approaches to finding a good computational grid. This topic has received a great deal of attention in the computational fluid dynamics community, and has been the subject of many conferences and reviews. It is extremely important for solving practical problems in realistic geometries or where complex flow patterns develop. Part of the problem is generating the initial grid, another part is modifying it appropriately as the flow or computational domain evolves. Localized improvements in resolution can substantially increase accuracy at relatively little computational cost.

The material covered in this chapter is complicated and covers information from many areas of research in a cursory manner. In fact, each section or even subsection deserves at least an extensive review article. For this reason, this chapter should probably be read again after reading Chapters 8 and 9.

6–1. Representations of Convective Flows

Chapter 2 introduced the concept of fluid convection and the physical complexities that arise in convection problems. Chapter 4 presented several straightforward finite-difference

algorithms for numerically convecting a fluid quantity across a grid. Because convection is nonlocal and involves large-scale motion, both phase and amplitude errors are equally important. This makes convection more difficult to treat accurately than local processes, diffusive effects, or wave and oscillatory phenomena. Because of these difficulties, methods used to represent and solve convection have a disproportionate effect on the numerical representations and associated gridding. Therefore, we first consider the special problems of representing convection.

6–1.1. Why Convection Is Difficult to Simulate

Even when the quantities being convected are all smoothly varying and well resolved by the chosen discretization, calculating the convection of a quantity ρ from one location to another usually demands more from the numerical techniques than calculating the effects of other processes. (Radiation transport, a possible exception to this, is discussed in Chapter 13.) Convection does not smooth fluctuations and gradients, therefore good algorithms for convection tend to lack numerical diffusion or the relaxation effects typical of local processes. Multidimensional convection usually has the opposite effect: it increases the amplitudes of short-wavelength components of flow structures. Shear in the fluid, for example, generally steepens gradients and generates more fine-scale structure as time advances. Thus short wavelengths as well as long wavelengths should be treated correctly by convection algorithms.

Convection is subject to the physical requirements of causality, positivity, conservation, and time reversibility. Therefore, these important qualitative properties should be built into numerical convection algorithms. In terms of a Fourier representation, both the phases and the amplitudes of the modes must be integrated accurately. When the flow is incompressible, distant fluid elements are strongly coupled, usually through an elliptic equation for the pressure. Correlating fluid properties over large distances to maintain incompressibility also introduces additional numerical problems with convection and boundary conditions.

The demands on convection algorithms are accentuated by the continually growing nature of phase errors. These errors become apparent when a structure, made up of components of many different wavelengths, is convected across a discrete computational domain. As shown in Chapter 4, errors in the amplitude of a given mode are bounded and may even decrease as long as the algorithm is numerically stable and the problem itself is physically stable. Phase errors, on the other hand, can increase indefinitely because they are not limited by stability or physical constraints, only by accuracy. For example, assume that the longest and shortest significant wavelength components in a structure have phase velocities that differ by only 3 percent. Then a flow structure incorrectly doubles in thickness by the time it convects a distance of about $\sim 1/.03 = 30$ times its original size. Furthermore, the profile is significantly distorted and unphysical undershoots and overshoots may appear in the numerical solution well before this.

Chapter 4 showed a computed profile departing steadily from the actual profile as the short and long scales in the computed approximation became uncorrelated. Diffusion can, in effect, reduce growing phase errors by damping out the short wavelengths. A subset of the numerical literature on convection is devoted to algorithms for *convection-diffusion problems*. When the physical diffusion is strong enough, these algorithms can take

advantage of the presence of physical diffusion to cover up the phase errors in numerical convection. Unfortunately, the amount of diffusion required to mask convection phase errors is usually orders of magnitude greater than the physical diffusion. Therefore, one of the major issues in computational fluid dynamics is how to minimize the residual numerical smoothing while controlling the unphysical effects arising as the sum of phase and amplitude errors.

6–1.2. The Continuity Equation

The continuity equation governs convective flow. It is often written in *conservation form*,

$$\frac{\partial \rho}{\partial t} + \nabla \cdot \rho \mathbf{v} = 0, \tag{6–1.1}$$

and $\partial \rho / \partial t$, often called the Eulerian derivative, is the rate of change of ρ at a point that is fixed in the coordinate system. There are other ways to write the continuity equation. The *convection form*, also called the *Lagrangian* form, is

$$\frac{\partial \rho}{\partial t} + \mathbf{v} \cdot \nabla \rho = \frac{d\rho}{dt} = -\rho \nabla \cdot \mathbf{v}, \tag{6–1.2}$$

where $\mathbf{v} \cdot \nabla \rho$ represents convection and $-\rho \nabla \cdot \mathbf{v}$ appears as a source term representing compression. The full derivative, $d\rho/dt$, used here is the Lagrangian derivative, the rate of change of the density ρ in the frame of reference that moves at the local fluid velocity. The *integral form* of the continuity equation is

$$\frac{\partial}{\partial t} \int_{\text{volume}} \rho dV = - \int_{\text{surface}} \rho \mathbf{v} \cdot d\mathbf{A}, \tag{6–1.3}$$

where dV is a volume element and $d\mathbf{A}$ is an area element. Equation (6–1.3) states that the amount of material $\int \rho dV$ in a volume can change only through flows of material $\rho \mathbf{v} \cdot d\mathbf{A}$ through the boundary of the volume.

These various representations of the continuity equation all make a statement about the behavior of the continuous variable $\rho(\mathbf{x}, t)$. Numerical approximations to ρ are projections of this infinite-dimensional vector onto a finite-dimensional subspace. When we approximate equation (6–1.1) numerically, we devise a means of using a finite approximation at some time t to predict the equivalent finite approximation to ρ at a later time $t' > t$. The inherent problems of representing a continuous function by a finite number of degrees of freedom have been introduced in Chapter 4 and are discussed further in Section 6–2.

The continuity equation, equations (6–1.1), (6–1.2), or (6–1.3), has several qualitative physical properties, mentioned above, which the numerical algorithms should reflect faithfully. One of these properties is causality: the material currently at a given point arrives there only by previously leaving another place and passing through all intermediate points. Another property is conservation: the total quantity of ρ in a closed system without source or sink terms does not change. In most situations, solutions found with nonconservative numerical methods show unphysical instabilities or growing errors. A third property is positivity: if the convected quantity ρ is originally positive, it will never become negative due to convection alone. Many of the common and annoying problems that arise in the

solution of coupled systems of convective equations occur because the quantities become unphysically negative when convected numerically.

The continuity equation is also time reversible: changing t to $-t$ and $\mathbf{x}$ to $-\mathbf{x}$ leaves the equation unchanged. This property is sometimes mirrored in the numerical methods, but it is usually ignored because even a small amount of physical diffusion breaks this symmetry. Galilean invariance is a related property that is difficult to include exactly in numerical solutions of fluid-dynamic convection. The use of any grid imposes, in effect, a preferred coordinate system on the numerical solution. This will affect both the permissible timestep and the residual numerical diffusion. Good numerical solution algorithms will tend to confine errors associated with different frames of reference to short wavelengths and small amplitudes.

As a basis for evaluating representations and the reactive-flow algorithms implemented in those representations, we list some general properties that the continuity equation algorithms should have. These algorithms should:

1. be numerically stable for all cases of interest,
2. conserve quantities that are conserved physically,
3. ensure the positivity property when appropriate,
4. be reasonably accurate,
5. be computationally efficient,
6. generalize to multidimensions, and
7. be broadly applicable, that is, not problem dependent.

The representations and algorithms described in this book are discussed in terms of these criteria.

6–1.3. Lagrangian versus Eulerian Representations

The convection of the variable ρ is governed by the appropriate continuity equation in conservation form, equation (6–1.1), or in Lagrangian form, equation (6–1.2). When ρ represents an *intrinsic* rather than a *conserved* quantity, the right side of equation (6–1.2) is zero. For example, an intrinsic quantity could be a ratio of hydrogen atoms to oxygen atoms, or a progress variable marking the location of an evolving interface. Whether to use an Eulerian or a Lagrangian frame of reference is an important choice.

In a discrete *Eulerian representation*, the spatial grid points are fixed, and parcels of moving fluid are convected across the grid from one cell to another. Such representations may be implemented efficiently and to high-order accuracy if the grid is uniform or only slowly varying. Convoluted interfaces in the flow cannot be localized more precisely than a single cell in an Eulerian representation without either introducing additional degrees of freedom in the representation or making assumptions about the extended smoothness of the solution. The most persistent problem arising in Eulerian representations is numerical diffusion, which moves a small amount of material across cells faster than any physical process. Numerical diffusion may appear as premature mixing throughout a computational cell, when, in fact, the mixing should have just begun at one interface or corner of the cell.

In a *Lagrangian representation*, the grid points and computational cells move with the fluid. Thus the continuity equation reduces to an ordinary differential equation for ρ in

the moving frame of reference. If ρ is a mass density composed of many species, each species density ρ_i is individually conserved as it moves with the fluid. Even when there are sound waves present, entropy is conserved along the fluid path lines when nonisentropic heating and cooling can be neglected. From the point of view of numerical simulation, there appears to be enormous simplifications and potential advantages to the Lagrangian representation. The equations for each individual element are simpler and interface locations can be tracked automatically. These advantages are counterbalanced by corresponding disadvantages. The cell interface geometry becomes more complex because the fluid elements can wrap around each other. This geometric complexity of cells and interfaces reduces numerical accuracy significantly and makes programming slow and difficult. The trade-off involves choosing between accuracy with geometric complexity in a Lagrangian representation, and computational simplicity with numerical diffusion in an Eulerian representation.

We have so far discussed Lagrangian representations in terms of small but finite volumes of the fluid that move, rotate, stretch, and interact with their neighbors. There are other ways to interpret Lagrangian representations. Another class of approaches interprets the fluid as being represented by a collection of *macroparticles*. This interpretation, and the corresponding numerical approaches, usually involve looking at slightly different physics or emphasizing the particulate aspects of the physics. These macroparticle approaches are an extensive subject discussed briefly in Section 6–2.5 and again in Section 9–6.

There is no easy answer to the question of which representation, Eulerian or Lagrangian, is better. A class of approaches, generally called arbitrary Lagrange-Euler (ALE) methods, have been developed over the last thirty years. These try to combine the best aspects of Eulerian and Lagrangian representations (Hirt 1970; Amsden, Ruppel, and Hirt 1980; Dukowicz and Kodis 1987). In ALE representations, a grid is established where the cells can move and distort, but the nodes and cell interfaces are not constrained to move only at the local fluid velocity. Thus the grid can track the flow in a Lagrangian manner in some regions but allows the fluid to cross the grid in other regions. In this way, the grid may retain enough structure to alleviate the worst of the timestep and grid distortion problems associated with fully Lagrangian representations (Benson 1992). ALE is often very useful for solving flows with sharp interfaces between fluids with different properties, such as those that occur between air and water or fuel and oxidizer. On the other hand, it often happens that the grid becomes so distorted in an ALE solution that further motion with the flow must be prohibited in extended regions. When this happens, ALE reduces to an expensive Eulerian calculation on a poorly structured grid.

Once a particular representation is chosen for a calculation, different types of algorithms and gridding procedures follow. Discussions of Eulerian, Lagrangian, and ALE representations will appear throughout the remaining chapters of this book.

6–2. Discretizing a Continuous Variable

The usual mathematical approach to discretizing a continuous variable $\rho(\mathbf{x}, t)$ is to expand it as a linear combination of a number of independent basis functions $\{\rho_j(\mathbf{x}, t)\}$,

$$\rho(\mathbf{x}, t) = \sum_{j=1}^{\infty} \rho_j(t) f_j(\mathbf{x}). \tag{6–2.1}$$

The coefficients $\rho_j(t)$ are usually functions of time only and the basis functions $f_j(\mathbf{x})$ vary with the spatial variable $\mathbf{x}$.

Accurate representations of even well-behaved, continuous variables require a large number of functions in an expansion. To make the representation computable, the number of discrete degrees of freedom, that is, the number of coefficients in the set $\{\rho_j\}$, must be moderate. Now medium-sized personal computers can easily handle two-dimensional representations with tens or even hundreds of thousands of degrees of freedom (coefficients). Computations solving three-dimensional models with a few million degrees of freedom are now relatively common, even on the workstations that are generally available to most users. This level of discretization, however, only allows spatial resolution of about one percent in each direction. To do better requires much larger expansions and correspondingly more computer resources.

6–2.1. Grid-Based Representations

In *grid-based* representations, physical space is broken into small regions, usually called *computational cells* or simply *cells*. This discrete representation then associates several numerical degrees of freedom and corresponding expansion functions with each cell. To be meaningful, the expansion functions $\{f_j(\mathbf{x})\}$ have to be reasonably localized in space, typically to about one of the cells. As soon as this association is made in the representation, and regardless of the formal order of accuracy of subsequent algorithms, an unphysical quantity – the cell size – is impressed on the numerical solution. If more than one degree of freedom is associated with each cell, spatial variations within a cell can, to some extent, be resolved. This reduces but does not remove cell-size effects.

Once a grid-based representation and discretization are chosen, a procedure to reconstruct the physical variables must be specified. This procedure generates a physically meaningful and consistent value for each physical variable at every point in the computational domain. When a linear expansion for the solution exists, as given by equation (6–2.1), that expansion can be used to define the reconstruction.

In an Eulerian approach, the expansions are generally related to a fixed *Eulerian grid*. Fluid dynamic variables are viewed as being transported or moved from cell to cell by fluxes that pass through the interfaces separating the cells. For this approach to be accurate, both the discretization and the reconstruction procedure must be able to represent smooth functions accurately. A Lagrangian approach, on the other hand, tracks the motion of a number of nodes that move with the fluid. This simplifies the equations by separating the problem into a geometrical part, tracking the configuration of the nodes, and a physical part, calculating how the physical properties of the fluid at each node change in time. For this approach to be accurate, the nodes always have to be clustered so that important local gradients and their properties can be approximated accurately. Connecting these moving nodes together in some sensible order related to their instantaneous locations gives another kind of grid, a *Lagrangian grid*. In Lagrangian representations, the flow itself determines the evolution of the grid. Thus Lagrangian representations are generally treated using only local expansions. Small-scale motions in the flow quickly result in significant local variations in a Lagrangian grid, making the influence of distant points difficult to determine accurately on the distorted grid.

Finite-difference, finite-volume, and finite-element algorithms (discussed in more detail in Chapters 8 and 9), are the most commonly used mathematical approaches for constructing solution algorithms in either Eulerian or Lagrangian representations. As described in Chapter 4, the values of the physical variables on a time- and space-discretized grid are computed at the end of the timestep from their values at the start of the timestep, and sometimes from a few previous timesteps. At each cell, time is usually updated to the same value as all the other cells so the concept of a time level for the entire spatial grid is meaningful. Thus the numerical treatment of time and space derivatives is usually quite different.

6–2.2. Eulerian Grid Representations

Consider a typical Eulerian representation consisting of a set of $J+1$ locations $\{x_{j-\frac{1}{2}}\}$ in a one-dimensional spatial domain bounded by $x_{\text{left}} \le x_{j-\frac{1}{2}} \le x_{\text{right}}$. These locations are ordered to form a *grid*, also called a *mesh* or a *lattice*, such that

$$x_{j-\frac{1}{2}} < x_{j+\frac{1}{2}}, \qquad j = 1, 2, \ldots, J. \tag{6–2.2}$$

The typical Eulerian grid, shown in Figure 6.1, divides the computational domain into J cells centered at $\{x_j\}$, where

$$x_j = \frac{1}{2}\left(x_{j-\frac{1}{2}} + x_{j+\frac{1}{2}}\right). \tag{6–2.3}$$

As can be seen in the figure, the original set of locations $\{x_{j+\frac{1}{2}}\}$ are the positions of the interfaces bounding the discrete computational cells. Other definitions are possible for connecting the locations of interfaces and cell centers. In spherical coordinates, for example, it might be better to choose a cell-center location that ensures equal volumes between $\{x_{j-\frac{1}{2}}\}$ and x_j and between x_j and $\{x_{j+\frac{1}{2}}\}$. Often the cell-center locations are not even needed by the solution algorithms. The cell sizes, $\{\Delta x_j\}$, are given by

$$\Delta x_j \equiv \left(x_{j+\frac{1}{2}} - x_{j-\frac{1}{2}}\right). \tag{6–2.4}$$

Representating a Function on a Grid

We consider three simple, discrete representations on this grid. The first specifies a real, finite-precision value ρ_j for each cell. In this representation, the sequence of values $\{\rho_j\}$ is

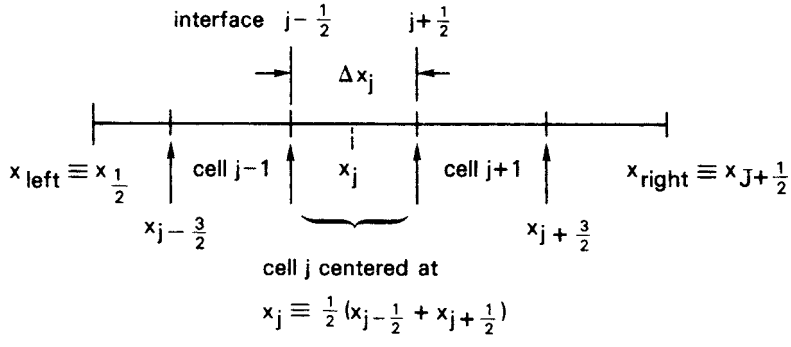

Figure 6.1. The computational domain of a typical one-dimensional Eulerian grid.

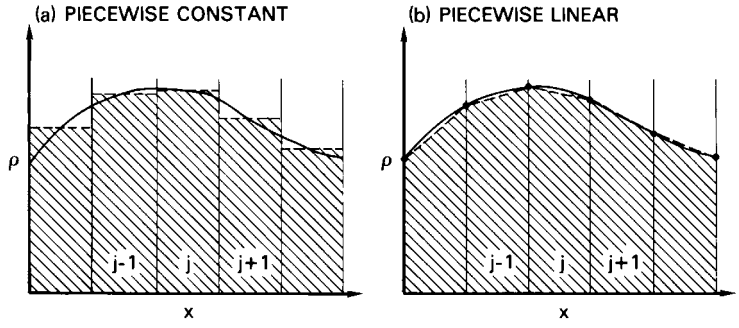

Figure 6.2. Two representations of a smooth function.

integrated to give the average value of $\rho(x)$ in each cell. Such a numerical reconstruction of the continuous variable $\rho(x)$ has discontinuities at each cell interface. Conservation of total mass M in the whole system is expressed as the sum

$$M = \sum_{j=1}^{J} \int_{x_{j-\frac{1}{2}}}^{x_{j+\frac{1}{2}}} \rho(x)\, dx = \sum_{j=1}^{J} \rho_j \Delta x_j. \tag{6–2.5}$$

This *piecewise-constant* representation of $\rho(x)$ is shown schematically in Figure 6.2a. The associated expansion function for this approximation is a localized *top-hat* distribution that is one cell wide and of unit height, shown in Figure 6.3a. The numerical approximation, the dashed lines in Figure 6.2a, can be written as a linear combination of local basis functions of this type. This reconstruction of the original continuous variable

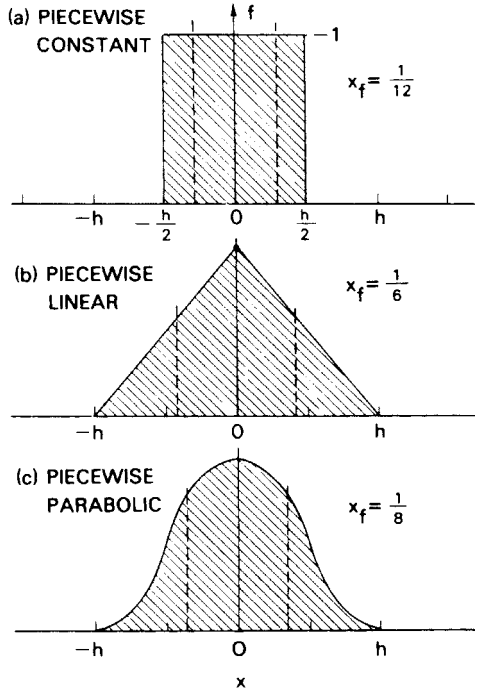

Figure 6.3. Expansion functions of three representations. The vertical dashed lines show the locations of x_f, the characteristic width of the three local expansion functions.

is not smooth. Nonetheless, higher-order accuracy is possible with the right choice of algorithm.

A *piecewise-linear* reconstruction results by assuming that the set of discrete values represent values of $\rho(x)$ at the cell interfaces. The reconstruction of values within each cell is then performed by interpolating linearly between the two interface values,

$$\rho(x) = \frac{\left[\rho_{j-\frac{1}{2}}\left(x_{j+\frac{1}{2}} - x\right) + \rho_{j+\frac{1}{2}}\left(x - x_{j-\frac{1}{2}}\right)\right]}{\Delta x_j} \tag{6–2.6}$$

for $x_{j-\frac{1}{2}} \le x \le x_{j+\frac{1}{2}}$. Conservation of total mass is now expressed as

$$\begin{aligned} M &= \sum_{j=1}^{J} \int_{x_{j-\frac{1}{2}}}^{x_{j+\frac{1}{2}}} \widetilde{\rho}(x)\, dx \\ &= \rho_{\frac{1}{2}} \frac{\Delta x_1}{2} + \sum_{j=1}^{J-1} \rho_{j+\frac{1}{2}} \frac{(\Delta x_j + \Delta x_{j+1})}{2} + \rho_{J+\frac{1}{2}} \frac{\Delta x_J}{2}. \end{aligned} \tag{6–2.7}$$

These two representations, piecewise-constant and piecewise-linear, have the same number of degrees of freedom and essentially the same mass integration formulas. The piecewise-linear representation of a curve, shown in Figure 6.2b, does appear to be more accurate than the piecewise-constant representation. The reason for this is related to the width and hence relative smoothness of the expansion functions used. The expansion function for the piecewise-linear representation is the *teepee*, *tent*, or *roof* function of unit height at one interface. This basis function drops linearly to zero at the edges of the two adjacent cells, as shown in Figure 6.3b.

Figure 6.3c shows an expansion function for one type of *piecewise parabolic* representation. Though this would seem to be a higher-order approximation than the piecewise-constant and piecewise-linear representations, the spatial extent of the underlying expansion function is actually somewhat less than that of the teepee functions. If the quantity x_f^2, the square of the extent of the basis function, is used to measure the effective width,

$$x_f^2 \equiv \frac{\int_{-\infty}^{\infty} x^2 f(x)\, dx}{\int_{-\infty}^{\infty} f(x)\, dx}, \tag{6–2.8}$$

the three local expansion functions shown in Figure 6.3 have

$$x_f^2 = \begin{cases} \frac{1}{12} & \text{piecewise constant} \\ \frac{1}{6} & \text{piecewise linear} \\ \frac{1}{8} & \text{piecewise parabolic.} \end{cases} \tag{6–2.9}$$

Thus piecewise-linear basis functions are actually the widest. The piecewise-parabolic basis functions, as defined here, are not as wide because linear terms have been left out of the expansion functions to keep them localized. When the linear terms and the quadratic terms are included, information from more than one or two adjacent cells is needed to determine the coefficients of the expansion basis functions. Such higher-order, less-localized representations are considered in more detail in Chapter 8 for solving continuity equations using finite-element and spectral methods.

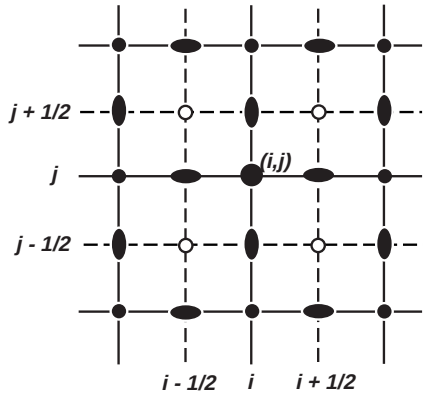

Figure 6.4. An example of a staggered grid. Values of the flow variables are defined on one of the grids and interpolated, as needed, onto the other grids. For example, ρ, $\rho\mathbf{v}$, E, $(\nabla \cdot \mathbf{v})$, P, T, and ϕ are defined at cell centers (marked by •'s), such as the point (i, j); v_x and $\nabla\phi_x$ are defined at the horizontal ellipses; v_y and $\nabla\phi_y$ are defined at the vertical ellipses; and $\nabla \times \mathbf{v}$ are defined at cell corners (○'s).

The top-hat basis function in the piecewise-constant representation is the closest that a discrete Eulerian representation can come to a δ-function, being just one cell wide. It is reasonable to expect that discontinuous fluid behavior such as shocks would be reproduced more faithfully by representations with minimal instantaneous transfer of numerical information over large distances in a single timestep, that is, by narrow basis functions. We also expect that smooth, potential, or incompressible flows would be well treated by discretizations with smooth, extensive basis functions, because the fluid behavior is globally coupled through the relatively fast acoustic waves.

Regular and Staggered Grids

In a regular grid, all of the variables are defined at the same location, for example, either at cell interfaces or cell centers. In staggered grids, different variables are defined at cell corners, cell faces, and cell centers. Figure 6.4 is a schematic of such a staggered Cartesian grid. Later in this chapter we describe how this idea is applied to unstructured grids to produce *covolume grids* or Voronoi-Delauney grids (Section 6–3.3).

Staggered grids are an important concept in computational fluid dynamics because they allow very natural definitions of variables (Harlow and Welch 1965). For example, in Chapter 9 there is an explanation of how defining the velocity components on the staggered grid leads to simple definitions of the difference operators for the divergence, curl, and gradient functions. One common difficulty with regular grid representations is that there can be unphysical, "spurious" numerical modes that appear as "grid decoupling" or "checkerboard" instabilities, and these appear even in fairly simple flow solutions (Arakawa 1966). Using staggered grids in many incompressible-flow or implicit algorithms seems to solve what is called the "spurious mode" problem (Nicolaides 1993). The cost, however, is the addition of some grid complexity in complex geometry.

6–2.3. Expansion and Spectral Representations

In practice, $\rho(\mathbf{x}, t)$ is represented as a finite, linear sum of expansion functions, $\{f_j(\mathbf{x})\}$, as shown in equation (6–2.1). The time-dependent coefficients $\{\rho_j(t)\}$ are determined initially from the starting density profile, and are then advanced in time by a numerical algorithm. When the basis functions are local, as in the three cases above, each of the J

time-dependent expansion coefficients can be associated with a local value or with an average value of ρ over a cell-sized region. The local value of ρ now multiplies this basis function in the expansion.

It is also possible to use the expansion in equation (6–2.1) with less localized eigenfunctions than illustrated in Figure 6.3. Finite-element methods are discussed in Chapter 8 and there are many excellent textbooks and reviews (see, for example, Zienkiewicz and Morgan [1983], Mohr [1992], and Morgan and Peraire [1998]). These methods use local expansions that couple all of the cells of the computational domain together implicitly. Information from the entire grid contributes to the evaluation of derivatives and to the integration of the expansion coefficients in time. As higher derivatives of the solution are matched at the element interfaces, both the number of degrees of freedom in the representation and the smoothness of the equivalent basis functions increases. Computing costs also increase. As the basis functions extend farther, it makes less sense to think of the coefficients as representing the values of ρ at the grid points or discrete node locations, and more sense to think of them as defining extended degrees of freedom.

Piecewise-cubic splines, for example, use a different cubic polynomial in each of the grid intervals. The four coefficients in the cubic expansion are chosen so that the composite spline function, its first derivative, and its second derivative are continuous at the interfaces of the cells. To ensure this smoothness, the problem must be solved implicitly, again coupling information across the entire grid. Even in explicit finite-difference formulas based on the piecewise-constant approximation, obtaining third-order accuracy in one dimension requires information from at least four grid points.

The assertion that higher-order accuracy, obtained by using more extended expansion functions, is better than lower-order accuracy can be carried to its logical limit. For example, the Fourier expansion used in Chapter 4 to represent and analyze continuous functions is entirely nonlocal. It uniformly weights information from over the entire spatial grid to construct the Fourier coefficients. In general, the method of weighted residuals, reviewed by Finlayson and Scriven (1966), uses two sets of functions, the actual expansion (basis) functions and a second set of weight functions that ensure that the basis satisfies the differential equations as closely as possible. Fourier or *spectral* representations (Gottlieb and Orszag 1977; Canuto et al. 1988) are distinct from finite-difference, finite-volume, and finite-element representations in that the expansion functions are infinitely differentiable global functions. The weight functions may be the same as the expansion functions (Galerkin spectral methods) or may be completely local, such as delta functions at a number of select points (collocation spectral methods). In either case, the integrated residual of the weighted expansion functions must be zero.

In the simplest Galerkin method applied to a periodic Cartesian domain, the basis is a discrete set of trigonometric functions, and the expansion coefficients are the amplitudes of Fourier harmonics. In these representations, it is conventional to think of a discrete grid in *k-space*, also called *wavenumber space*, rather than an *x-space* or *configuration-space* grid. Because the Fourier harmonic is an average containing information from all locations with the same magnitude weights, it is the antithesis of a local value of a physical variable. For studies of large-scale collective phenomena or other phenomena where information can be assumed to propagate everywhere at essentially infinite speed, Fourier representations are excellent methods to use. Acoustic pulses, flame fronts, and detonations do

propagate at finite speeds. There is a danger in using spatially extended expansion functions for such discontinuous and localized phenomena: nonlocal expansion methods allow the numerical solution in a region to respond to an approaching discontinuity before the discontinuity actually arrives. This artifact of wide expansion functions is not physical and often produces unacceptable results. Using such nonlocal expansions to represent a gas dynamic-shock problem may result in solutions with unphysical ripples appearing at the upstream boundary. This occurs because the methods implicitly couple the solution throughout the grid. The speed of information propagation in the numerical representation of the medium is then unphysical.

Between very localized and global representations, there are families of expansions that are progressively more delocalized in the sense that they use more values from cells that are further away. For example, there are the intermediate representations called *spectral elements* in which the expansion degrees of freedom are associated with variably shaped patches in the computational domain (Patera 1984; Canuto et al. 1988). Another expansion approach, *wavelets*, is an area of very active research (see, for example, Benedetto and Frazier [1994], Beylkin and Keiser [1997]). As with spectral elements, the idea is to employ an expansion where there are a number of expansion functions for each region, each representing a different scale length in the solution. Again, the expansion functions depend on both k and x. Unlike spectral elements, however, the expansion functions are not grouped in patches. Thus they can be dynamically adapted to better represent the evolving flow.

The trade-offs involved in choosing a representation involve balancing the aspects of local and nonlocal representations. Local representations seem better suited to discontinuous phenomena and more easily retain important aspects of finite information propagation speeds. Very smooth functions are more easily treated and integrated accurately with nonlocal methods. Local methods appear to have an advantage when complex geometry is considered, as discussed in Sections 6–3 and 6–5.

6–2.4. Lagrangian and Unstructured Grid Representations

A different approach to the problem of representing a continuous function is to identify finite regions of the fluid, called *fluid elements*, and to convect them as separate entities. In this Lagrangian approach, the physical properties of a moving fluid element are tracked. Conservation laws are then formulated in terms of adjacent, interacting fluid elements. A particular atom in the fluid, for example, is associated with only one Lagrangian element and should only pass through a cell interface from one element to another by physical diffusion. Lagrangian representations are attractive because they appear to encompass the maximum possible localization in order to maintain sharp material interfaces without numerical diffusion.

Lagrangian representations include a broad range of methods that track moving fluid elements and treat macroparticles. At the end of a timestep in a Lagrangian calculation, the physical variables are integrated and the locations $\{\mathbf{x}_l\}$ of the fluid elements or particles are updated according to

$$\frac{d}{dt}\mathbf{x}_l(t) = \mathbf{v}_l(t). \tag{6–2.10}$$

These moving node locations become degrees of freedom in the solution because they change every timestep. For a given spatial resolution, a Lagrangian representation appears to have extra degrees of freedom relative to Eulerian representations because the locations of the nodes, as well as the dependent variables, are free to vary.

Consider the motion of a small cloud that moves through the surrounding fluid without diffusion. Suppose that we represent the cloud in a two-dimensional simulation by a number of cells, each containing a mass m_l at a location $\mathbf{x}_l$. If a shear flow is imposed across the center of the cloud, the adjacent cells distort and move as the flow evolves. No fluid can cross Lagrangian cell boundaries because these boundaries are assumed to be moving with the flow. After a time, Lagrangian nodes that were initially close together may no longer be close, and some nodes that were initially distant begin to approach each other. When general rotational motions or shears are important in flows, a Lagrangian grid tends to become tangled and soon needs continual restructuring. This is shown in Figure 6.5 for a shear flow induced by a Rayleigh-Taylor instability.

Restructuring a grid such as the one shown in Figure 6.5 requires either a remapping procedure that moves the cell interfaces relative to the fluid, or changing the grid connectivity to allow each Lagrangian node to have different neighbors as the flow evolves. The first process is numerically diffusive. The second process is not necessarily diffusive and has been implemented on dynamically restructuring triangular and tetrahedral grids (Fritts and Boris 1979). Lagrangian methods are discussed further in Section 6–4 on adaptive gridding, and again in Chapter 9.

6–2.5. Macroparticle Representations

The reactive-flow equations given in Chapter 2 describe the continuum fluid limit of a system controlled by collisions between a vast number of atoms and molecules. It is natural to ask whether there may be useful computational representations of a fluid based on the particulate nature of matter. Indeed, from the earliest work in computational fluid dynamics, it was understood that there are Lagrangian aspects of fluid motion that are well reproduced by tracking cohesive entities of macroscopic parcels of fluid, which are called *macroparticles*. The parcels were convected as single, large particles. Such *macroparticle* representations are an important class of Lagrangian representations.

Macroparticles move at the local fluid velocity defined as a suitable average of all the nearby macroparticle velocities. Macroparticle superposition determines the local value of the continuum density. Two or more of these macroparticles may overlap in a volume of space, unlike the usual Eulerian and Lagrangian cells. By releasing a large number of macroparticles into the simulation volume, the density in any region can be approximated by counting all the particles nearby and taking their respective masses into account. Each macroparticle actually represents many thousands, millions, or billions of real particles. This is the case for the direct-simulation Monte Carlo method (Bird, 1994) and particle-in-cell (PIC) method (see, for example, Monaghan [1985, 1988]; Brackbill and Ruppel [1986]) used for flows in which the mean free path between collisions is relatively long.

Values of position, velocity, and additional degrees of freedom, such as temperature and density, are assigned to the macroparticles. A recent class of such methods called *smoothed-particle hydrodynamics* (SPH) (Monaghan 1985, 1988; Swegle et al. 1994) is

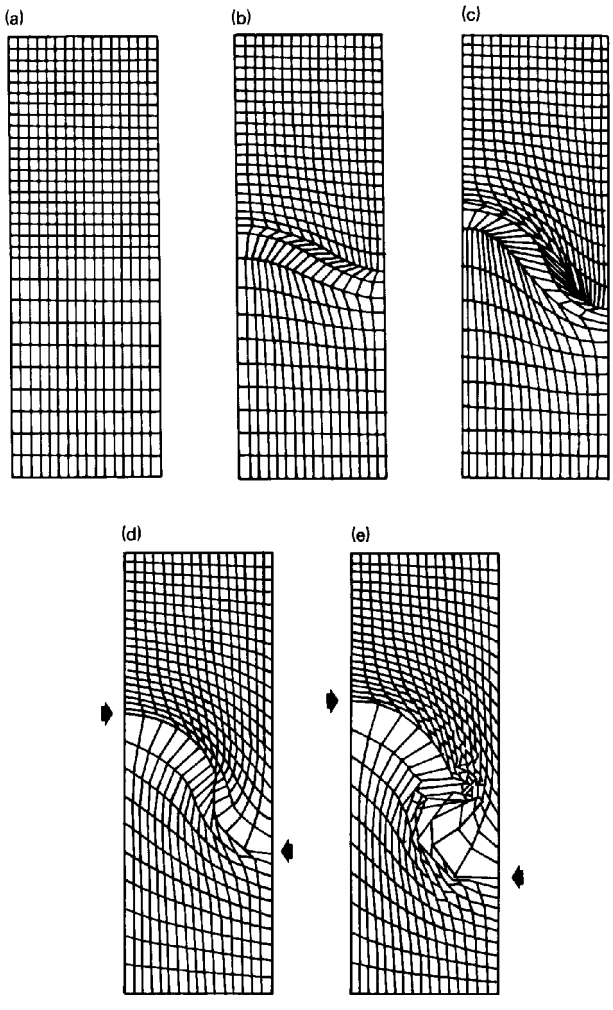

Figure 6.5. A Lagrangian calculation of a Rayleigh-Taylor instability on a quadrilateral grid. A heavy fluid is on top of the light fluid, and the effects of gravity are included. A small perturbation at the interface at the beginning of the calculation initiates the instability. The arrows on (d) and (e) indicate the location of the interface on each side of the figures. By (e), the calculation cannot proceed because grid lines have crossed. To continue the calculations, some regridding procedure must be carried out. (Courtesy of M. Fritts.)

described briefly in Chapter 9. Macroparticle methods do not generally rely heavily on an underlying grid. Continuum fluid properties, however, are often defined on an auxiliary grid by interpolating and averaging the properties of nearby macroparticles. Because the use of this auxiliary grid is controlled by the specifics of the macroparticle algorithms, it is generally fruitless to discuss the representation for these methods independently of the algorithms.

There is one class of macroparticle models in which there very definitely is a grid. *Lattice-gas* (LG) methods extend the particle models to their logical limit (Frisch, Hasslacher, and Pomeau 1986; Simon 1993). The grid or *lattice* is perfectly regular and

the particles in the lattice gas move only along the discrete links between the fixed nodes in this lattice. Not only are the locations of the particles discretized, but the velocities are discretized as well. For many flows, it has been shown that the solutions of such a discrete, efficiently computable system reduce rigorously to the Navier-Stokes equations as the number of particles becomes large (see the review by Chen and Doolan [1998]). The lattice gas is a model physical system with many rigorously provable properties (Boon 1991). As with SPH and other macroparticle methods, the lattice-gas representation is totally tied to particular solution algorithms.

Macroparticle methods have advantages over grid-based approaches for solving the convection equations because they can easily mirror the mass-, momentum- and energy-conserving properties of a reactive flow. Positivity is also easier to guarantee. Numerical stability and accuracy of particle trajectory integrations are well understood and the solution algorithms are now very efficient. The main problem with macroparticle approaches is their statistical nature. The number of macroparticles that can be used, even on the largest supercomputers, is many orders of magnitude smaller than the number of atoms or molecules in even a small volume of a real reactive flow. As a result, the derived quantities are subject to large statistical fluctuations. In reactive flows with extremely sensitive chemical processes, these fluctuations can be particularly disruptive. It is also difficult for macroparticle methods to reproduce, for example, a linear fluid instability growing smoothly over many orders of magnitude. As a result, macroparticle methods can be inefficient and expensive since more particles means better answers. Further, the sums and interpolations required to compute continuum quantities (such as temperature, density, and pressure) at any point in space often give discontinuous and computationally inaccurate values. Finally, there is an issue involved with evaluating derivatives, which involves using values of physical variables at localized subsets of the particles in the representation. As the flow evolves, the derivative subsets change, and there can be differences in the values of the derivatives. What results is a source of finite fluctuations in derivatives from infinitesimal changes in particle positions.

The prescriptions for how macroparticles move, the forces acting on them, and the way continuum fluid properties are derived from the properties and locations of all the macroparticles lead to a number of different algorithms. Several of these are discussed in Chapter 9.

6–3. Gridding to Represent Complex Geometry

Finding a suitable grid to represent a realistically complicated multidimensional geometry has always been a roadblock in computational fluid dynamics. The geometrical complexity of a multidimensional computational domain usually means that both the grid and the flow are complex. The gridding difficulties are independent of the particular finite-difference, finite-volume, or finite-element algorithm chosen. Here we discuss generic gridding approaches and the trade-offs involved in using them.

In general, more can be gained by intelligent gridding than by extending the integration algorithms to higher order. This is because high resolution is usually not needed everywhere, and lower-order formulas are generally more robust, are easier to program and use, and require less memory. Thus there has been much work to develop the art

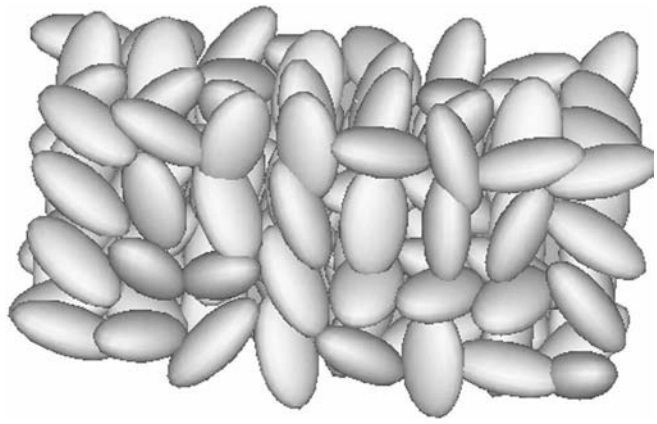

Figure 6.6. A very complex flow geometry generated by randomly placing identical ellipsoids. The surface of each ellipsoid has a random, high-frequency, low-amplitude perturbation added to model surface roughness.

and science of *grid generation*. Grid generation has been the subject of many conferences, review articles, and books (for example, Eiseman [1985]; Thompson, Warsi, and Mastin [1985]; Castillo [1991]; George [1991]; Knupp and Steinberg [1993]; Schneiders and Bunten [1995]; Thompson and Hamann [1997]).

Figure 6.6 shows an idealized but still complex geometry generated to compute the flow through a pile of grains with rough surfaces. A number of similar hard ellipsoids are placed in random locations and orientations. This kind of complex geometry might be used, for example, to simulate problems in fluidized-bed combustion or chemical vapor infiltration (a manufacturing process for high-performance composite materials). To illustrate various approaches to gridding, we extract a small portion of this porous medium and idealize it further to two dimensions. Two ellipses overlap, creating a nearly orthogonal corner on one side and a sharp acute corner between the ellipses on the other. The four parts of Figure 6.7 show the major aspects of (a) block-structured, (b) chimera and overset, (c) unstructured (finite-element), and (d) global Cartesian grids.

The simplest grids in multidimensions are constructed from sets of parallel grid lines (or planes in three dimensions) intersecting at right angles to form a global Cartesian grid. This is shown in the background of Figure 6.7b and the foreground of Figure 6.7d. The resulting computational cells are rectangles. The rectilinear grid lines in multidimensions define a locally orthogonal grid, for which many efficient standard solution algorithms apply. The problem with this approach is apparent from the figures. Except in the luckiest of circumstances, structures in the flow domain are not aligned with the grid. In Section 6–3.4, we consider techniques that allow accurate computation even though the grid is not aligned with the bodies. First, however, consider three approaches in which the grid is fit to the geometry of the body.

6–3.1. Eulerian, Body-Fitted Grids

A very good tutorial of the processes involved in transforming a physical grid from standard Cartesian (x, y, z, t) coordinates into a more appropriate $(\zeta, \eta, \psi, \tau)$ computational

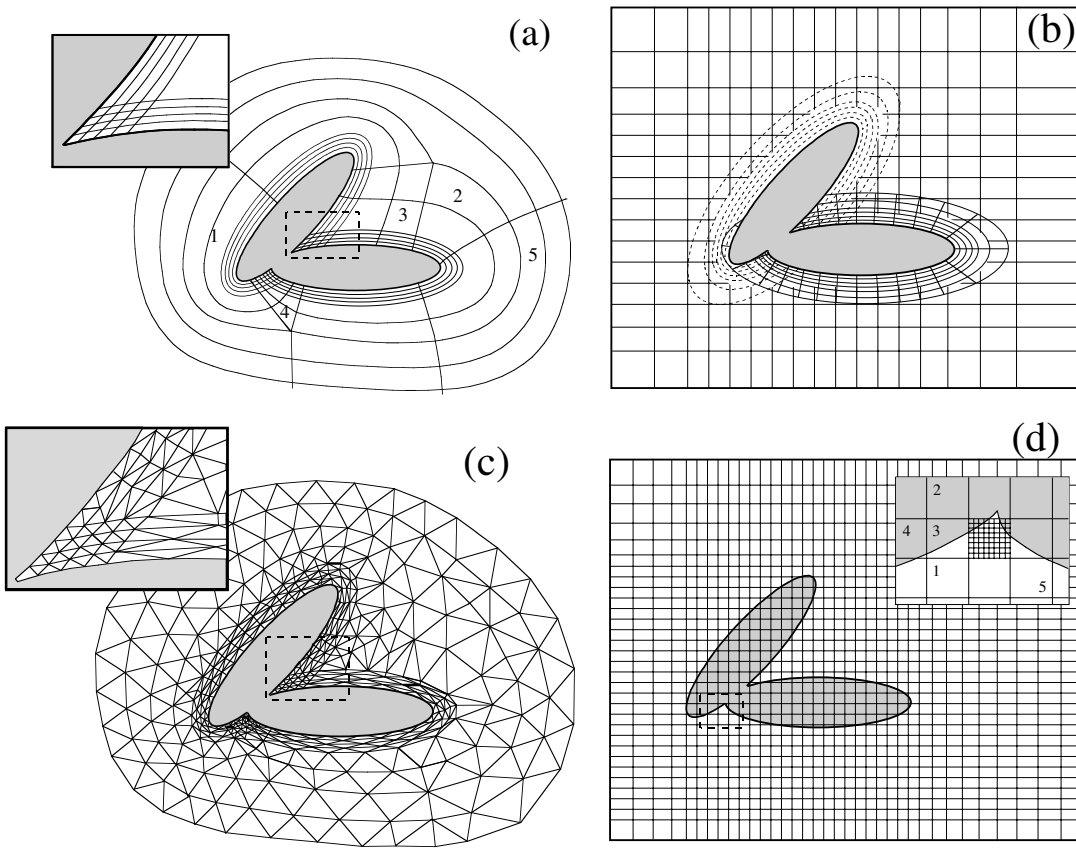

Figure 6.7. Schematic illustration of four approaches to gridding a segment of the complex geometry in Figure 6.4. (a) Block-structured grid. (b) Overset grids. (c) Unstructured triangular grids. (d) Global Cartesian grid.

coordinate system is given by Anderson (1995) and Tannehill, Anderson, and Pletcher (1997). This development involves recognizing where the internal boundary surfaces are, using these to define the coordinate axes, and developing the coefficient metrics and Jacobians required for the coordinate transformation of the equations to be solved. This may be a straightforward though tedious process, and some care is required to evaluate the coefficients correctly for each particular problem. The rewards for this are grids on which a fluid computation can represent the physics of the problem much more accurately than a standard, evenly spaced Cartesian grid.

Figure 6.7a shows a block-structured grid around the two ellipses. The composite grid has five distinct but aligned blocks, identified by numbers in each block. The outer block 5 wraps all the way around the composite body. The block can be extended outward by adding more layers of cells. Such a circular block is sometimes call an *O-grid*. Blocks 1 and 2 are often called *C-grids* because of their shape as they wrap around the body. Blocks 3 and 4 are rectangular grid patches. Each block is topologically rectangular, being formed by two distinct sets of grid lines. In three dimensions, the term "structured" extends to methods in which the grid sections are topologically cubical and the indexing of the cells within the separate domains is simple and regular.

The grid lines are not generally orthogonal in any of these blocks, although an orthogonal grid is possible in this two-dimensional case. Away from the bodies, only a few grid lines are shown. This approach, using adjoining distinct blocks, allows mesh lines to be aligned with the body surface and permits fine gridding adjacent to bodies. Some of the grid lines radiating away from the body change direction at block interfaces away from the body. Although any particular block in such a structured grid may appear quite irregular, the connectivity of adjacent cells is totally determined by the grid indices, lending significant simplification and efficiency in programming algorithms. In Chapter 9, a block-structured grid representation for free Lagrangian and macroparticle methods called the *monotonic Lagrangian grid* is briefly described. This algorithm can be used to show how to construct a structured grid block in index space through an arbitrary set of node locations (Boris 1986; Lambrakos and Boris 1987; Sinkovits, Boris, and Oran 1994).

Regularity and simple block structure can lead to improved accuracy in the solution algorithms. Transforming the partial differential equations being solved to regularly gridded index coordinates allows more accurate treatment of smooth body surfaces where the surface is curved between the nodes of the discrete grid. Derivatives evaluated by differences in the transformed space automatically take the curved grid lines into account. The price for this is twofold: the formulas are more complex because of the transformations between grid index space and real space; and enforcing strict conservation of mass, momentum, and energy through the transformation is difficult. (This approach should be contrasted to the unstructured-grid approach described in Section 6–3.3. In that case, the body is treated as a number of flat panels meeting at edges where the surface normals may be discontinuous.)

In a block-structured grid, the grid spacing along the body surface is generally much larger than normal to the surface, so the fine cells near the body have high aspect ratios. While this is computationally efficient, accuracy can suffer. For example, if a shock or flame hits the body at an angle to the grid lines, the effective resolution will be the coarser of the two resolutions, not the finer. Problems can also occur at acute corners of the type shown in the inset to Figure 6.7a. Nevertheless, the body surface can be treated as a true smooth curve, even between the grid lines. This property is not so easily duplicated in the unstructured gridding or in the globally structured Cartesian gridding approaches described in Section 6–3.2.

Constructing a block-structured grid from an arbitrary body geometry is a complicated task to which an extensive literature has been devoted. Thompson and Hamann (1997) describe the main methods and available software packages for implementing these tasks. The grid generation for body-fitted grids has two main stages: (1) determining the number, shape, and alignment of the blocks and then (2) determining the node locations within each of these blocks. The first of these stages is relatively difficult to automate. The second stage is more easily automated and involves either some form of interpolation within each block or the solution of a set of hyperbolic or elliptic partial differential equations. Grid quality, particularly at block interfaces and near corners and body surfaces, can have a strong impact on accuracy and so must be monitored.

6–3.2. Overset Grids

Figure 6.7b illustrates a composite grid generated by the overset-gridding method (Meakin and Suhs 1991, 1995; Maple and Belk 1994). The original implementation of this approach

was called *Chimera* gridding by Benek and Steger (Benek, Steger, and Dougherty 1983; Benek, Buning, and Steger 1985; Benek, Donegan, and Suhs 1987). In Figure 6.7b, separate grid patches are fit completely around each ellipse. Creating a good grid at ellipse intersections is not the problem it was for block-structured grids because grid patches are now allowed to overlap. Further, the interfaces between different patches on the overlapping grids do not have to be aligned.

In block-structured gridding, a single numerical solution is continued from one patch to its neighbors. In overset gridding, interpolation procedures are used to move information among separate parts of the solution computed on different grid patches. The grid is aligned with the body because at least one grid line lies along each body surface. In this particular two-dimensional case, the individual grid patches can be made orthogonal (although this is not done exactly in Figure 6.7b). As with the block-structured grid in Figure 6.7a, there is high resolution in the direction normal to the surfaces of both ellipses. To make the figure easier to read, the outward grid lines from the ellipse on the left are omitted from the figure. They are present on the lower ellipse, showing that anisotropic resolution is possible near body surfaces.

The solutions are computed separately on all the grid patches for a timestep or for one solution iteration, and interpolation from one patch to another defines the boundary conditions for the grid blocks. Computations on the two local grid patches around the bodies in the figure are used instead of values determined from the background grid. Therefore, the excluded region of the background grid is indicated as a "hole" in the figure where the Cartesian grid lines are interrupted near the bodies. The computations may or may not actually be performed on the excluded cells in the Cartesian background grid near and within the bodies, depending on efficiency and programming simplicity. The information associated with updating these cells is not needed because better answers are available from one or the other of the elliptical grids.

Overset-grid representations can be particularly efficient on parallel computers because the solution of the governing equations on each block can be performed separately by distinct processors. Interprocessor communication can be restricted to the necessary interpolation operations between patches. Load balancing among the processors is optimized when the grid patches have approximately the same number of cells, which can be achieved both by dividing up large patches and by consolidating small patches. Furthermore, different solution algorithms can be used on different patches. In Figure 6.7b, the two body-fitted, elliptical grids are shown embedded in a global Cartesian background grid. Simple, accurate, and particularly efficient algorithms can then be used over most of the domain where the grid is regular and Cartesian. For example, an implicit method can be used normal to the body surfaces in the patches that define the body while a fully explicit method might be used in the background grid.

The overset-grid approach is also effective when two bodies are moving relative to each other. Only the interpolations in the shifting overlap region have to change as one grid patch moves relative to another. The representation of the complex bodies as overlapping grid patches can remain fixed, and the grid patches themselves can be rigid. Thus overset methods have been used to simulate problems with objects ejected from moving bodies, such as a pilot ejected from a plane or boosters dropping from multistage rockets (Meakin and Suhs 1989; Tramel and Nichols 1997), and rotating blades in turbines, compressors, and rotors (Duque 1994).

Grid generation for overset grids is also a two-stage process. The first stage, breaking a geometry down into overlapping grid patches, is much easier to automate than generating a block-structured grid for a complex body configuration. The second stage, determining a hierarchy for performing the interpolations between the various grid patches, is more difficult. The successful automation of this procedure (Meakin 1991) has made this class of grid representations relatively easy to use. The interpolation procedure is a source of problems because it is difficult to ensure conservation of mass, momentum, and energy in the fluid moving between the distinct grids. Now there are several approaches for generating conservative interface algorithms for overlapping grids (Chesshire and Henshaw 1994; Wang 1995).

6–3.3. Unstructured Grids

Figure 6.7c shows an unstructured grid composed of triangular cells. These cells are tetrahedra in three dimensions. Such grids are flexible because an arbitrary number of triangles or tetrahedra can meet at each vertex. The grid is aligned with the body surfaces, and resolution can be enhanced near the surfaces and reduced in regions away from the surfaces. The inset in the figure shows that the grid can treat body intersections. Unstructured-grid representations can allow combinations of tetrahedra, prisms, and hexahedra (see, for example, papers in Crowley, Fritts, and Trease [1985]). This generalization increases a grid's generality, but also increases the program complexity.

Computing a derivative on an unstructured grid requires using nearby values of the fluid-dynamic variables that surround the point where the derivatives must be evaluated. For this reason, the quality of the unstructured grid affects the accuracy of the solution. For example, Figure 6.7c shows a moderately good grid throughout most of the domain. The gridding is, however, poor at the body surface near the lower ellipse. The presence of long, flat triangles adjacent to the body, with no triangle edges perpendicular to the body, means that sharp normal gradients are not resolved. Such gradients could be important in representing boundary layers. On the ellipse on the left, the radial grid near the surface allows normal derivatives to be evaluated accurately.

Unstructured grids have a long history as the representation underlying finite-element solution algorithms (see Section 8–5 and review by Morgan and Peraire [1998]). Over the last twenty years, use of these grids has been extended to the solution of fluid-dynamic problems. In the 1970s, unstructured grids of triangles were implemented for fluid dynamics (Crowley 1971), magneto-hydrodynamic (MHD) algorithms (Brackbill 1976), and adaptive triangle-based meshes (Fritts and Boris 1979). Voronoi meshes (Peskin 1977), in which the computational cell is a polygon formed by connecting the centroids of triangles, were the bases for a number of approaches, some of which are discussed in Chapter 9.

Unstructured grids first received attention in fluid dynamics in conjunction with Lagrangian models, in which the cells or fluid elements stretch and distort as the flow progresses. As the fluid elements become tangled in rotational or shear flows, the grid geometry rapidly becomes very distorted. The only really effective procedure to handle such distortions is to reconnect the moving nodes into a new grid, thus altering the *grid connectivity*. Unstructured grids make this Lagrangian reconnection possible because they are less constrained geometrically and topologically than the block-structured grids. Grid

reconnection, however, is still difficult to carry out in a reliable, automatic way, particularly in three dimensions. Nevertheless, there are advantages to reconnecting the nodes, and substantial effort has gone into developing reconnecting, adaptive Lagrangian grids. Unstructured-grid representations have been central in this activity because the number of triangle or tetrahedron edges meeting at a vertex is not fixed, and may vary during grid reconnection procedures.

General-connectivity grids change with the flow patterns during the course of a calculation. Some of the early efforts on multidimensional Lagrangian grids are summarized in the conference proceedings edited by Crowley et al. (1985). In the last few years, there has been major progress for Eulerian and Lagrangian calculations. ALE techniques, originally developed for structured quadrilateral grids in Lagrangian models (Hirt 1970; Dukowicz and Kodis 1987), are finding application to unstructured grids as well. Applications now include finite-difference and finite-element calculations of classical fluid-dynamic instabilities, acoustic- and electromagnetic-wave propagation, heat conduction, water-wave interactions with structures, impact deformations, the oscillation and breakup of droplets, and hydrodynamic problems for both compressible and incompressible fluids.

ALE gridding allows for more flexible motion of the nodes than either strictly Lagrangian or Eulerian gridding. When the motion of the nodes defining the grid is not fixed and not tied to the fluid, more general approaches can be considered that use the node (grid) motion in a constructive way. In principle, the use of finite-element algorithms can minimize errors by adjusting the location of the grid nodes. This means that additional equations are solved along with the fluid variable equations to move the nodes optimally. Section 6–4.5 on adaptive mesh refinement for unstructured grids describes how new nodes are added to improve resolution.

Despite their recent successes, unstructured-grid algorithms cost more per grid point to integrate than structured and block-structured methods. Because they are also more complicated to implement, their use on parallel computers has lagged accordingly. Constructing an unstructured grid for a complex geometry should be better suited to automatic procedures than generating block-structured grids. In principle the problem should be largely one of combining algorithms "that are automatic robust, and yield suitable element shapes and distributions for the flow solver" (Mavriplis 1997). In practice, the global topological difficulties and constraints imposed by block-structured grids are replaced with complicated and relatively inefficient local logic and with problems stemming from the possible interpenetration of logically remote tetrahedra that physically should not interpenetrate.

There are three general procedures used to generate unstructured grids: Delauney triangulation, tree methods, and advancing front methods (Thompson and Hamann 1997). Delauney triangulation results in a unique grid, given a set of nodes, in either two or three dimensions. This grid has been studied extensively because it has some rigorous properties relevant to evaluating accurate derivatives on the resulting grid. Every triangle or tetrahedron has nodes that lie on the surface of a sphere. In the Delauney interconnection, no other nodes in the system lie within this "circumsphere." In other words, for each tetrahedron, there is a point in space that is equidistant from the four defining nodes and no other node is closer. There is only one way of connecting the nodes into tetrahedra that accomplishes this. The Delauney construction also has a second grid, the *dual grid*,

composed of cells around each node of the triangulation. The volume associated with each node is the volume that is closer to that node than to any other node.

That the Delauney grid is unique is a mathematical fact, but it also has two practical drawbacks. Parallel algorithms to find the Delauney grid are difficult to devise. Furthermore, small changes in the location of nodes usually occasion significant changes in the connectivity of the grid. Even worse, physically meaningful boundaries in the geometry (such as the surface of a combustor nozzle) may not correspond to a face of one of the tetrahedra required by the construction. The only remedy is to modify the locations of the nodes or add nodes to describe the surface triangles.

Unstructured gridding methods based on quadtrees (two-dimensional trees) and octrees (three-dimensional trees) begin with a very coarse grid that spans the domain to be solved. The gridding procedure then hierarchically subdivides this coarse grid as required to obtain the desired resolution in any particular region. With unstructured grids, complex geometry, or an expected complex flow solution in one region, is treated by successively splitting the parent tetrahedra from the initial coarse gridding. Typically tetrahedra are split into four small elements (see, for example, Löhner and Baum [1990]). The method associates nodes on the defining surfaces with nodes on the tree being constructed to represent the geometry of the volumetric grid. This approach for constructing the initial grid is efficient for unstructured and structured grids alike (Yerry and Shephard 1985) and is closely allied with the techniques used for adaptive mesh refinement. A good exposition for Cartesian adaptive grid generation is given by Young et al. (1991).

In the advancing-front method, the surfaces of all the bodies are first triangulated. Then new nodes in space are defined just off this surface by projecting outward from the individual triangle faces. These nodes are connected to form a new surface standing just off the body surface, at which point the procedure can be repeated. Depending on the node distances and angles, existing nodes from adjacent advanced surfaces may be connected to form new tetrahedra without generating additional nodes (Löhner and Parikh 1988). This is a very fast method, but it can become complex when advancing fronts that originate from different surfaces begin to intersect. There is no local test that can be performed to know that two fronts are approaching each other, for example from the opposite sides of the domain. As a result, interpenetration of tetrahedra is possible and the evolving grid must be repeatedly checked for this.

6–3.4. Globally Structured Cartesian Grids

Uniformly spaced Eulerian grids, such as those that appear in the central regions of Figures 6.7b and 6.7d, are the simplest type of multidimensional grid. When a Cartesian grid is nonuniformly spaced, often to reduce the cost of calculations in regions where resolution is not a problem, the grid lines are still orthogonal and computation can be very efficient. In Figure 6.7b, a regular Cartesian grid is used as a background grid over which other body-fitted grids are superimposed. In Figure 6.7d, the entire grid is Cartesian and regular. It passes into the solid bodies, and the grid stretching is illustrated away from the body. This is a significant simplification in the grid representation.

Three different types of cells can be identified in a regular Cartesian grid: (1) fully open cells that are completely in the fluid, (2) fully obstructed cells that are completely

in the body or bodies, and (3) partially obstructed cells that intersect the body surface. An example of each of these is indicated as 1, 2, and 3, respectively, in the inset to Figure 6.7d. Special algorithms have been developed to treat the partially obstructed cells. Berger and LeVeque (1989) demonstrated the concept in two dimensions and showed that special treatment of only the body-surface cells, without changing the grid, could remove the unphysical, grid-scale "staircasing" effect caused by using a Cartesian grid to resolve smooth body surfaces. The method has since been extended to three dimensions for complex geometry (Landsberg et al. 1993; Melton et al. 1995; Aftosmis, Melton, and Berger 1995; Aftosmis, Berger, and Melton 1997) and to efficient parallel processing implementations (Young, Landsberg, and Boris 1993). Today, global Cartesian approaches for gridding complex geometry in use are chiefly differentiated by their treatment of the obstructed cells. (See, for example, Landsberg et al. [1993, 1995], Quirk [1994], and Aftosmis et al. [1995].)

Virtual Cell Embedding for Accurate Cartesian Grids

Virtual cell embedding (VCE) is a Cartesian gridding technique used for generating grids for very complex boundaries (Landsberg and Boris 1997). It differs from other Cartesian gridding methods for complex geometries in several ways. First, it only recognizes one general type of obstructed cell. Second, it uses a very simple technique to input the complex geometry. Finally, it computes the surface area and volume factors needed in the flow solver by a simple counting procedure. These three choices are all coupled to simplify the geometry specification and grid generation.

In VCE, a complex geometry is specified as the union of a number of simpler shapes. Each shape, whether given by analytic functions, a surface-panel representation, or some form of bit map, must be accompanied by a subroutine that determines whether any particular point $\mathbf{X} = (x, y, z)$ lies within or outside any of the bodies. In this representation, any number of distinct shapes can touch or overlap arbitrarily, as shown in Figure 6.6. The inset in Figure 6.7d shows how VCE works for the lower ellipse intersection, identified in the figure by the dashed box. The cell labeled 1 in the inset is entirely in the fluid and treated by whatever simple efficient algorithm applies to free space. The cell labeled 2 is entirely inside the body and no integration need be performed there.

Cell 3 lies on the surface of the body; some of its volume is in the fluid and some in the body. The cell just to the right of cell 3 is subdivided into an 8×8 grid of subcells. If the point at the center of a subcell is in the body (eleven of them are), the entire volume of the subcell is treated as solid body. If the subcell center is in the fluid (fifty three of them are), the whole subcell volume is counted as accessible to the fluid. Thus a simple counting procedure can be used to determine cell partial volumes. This can be performed to whatever accuracy is necessary to match the numerical accuracy of the chosen convection algorithm on the larger cells of the open grid.

Partial cell interface areas are found the same way. In this two-dimensional example, the cell volumes are really areas and the interface areas are really lengths. For example, the cell interface between cell 3 and the cell showing subcell gridding just to the right of cell 3 is composed of eight line segments, three of which have centers inside the elliptical body on the upper left. Thus $5/8$ of this interface area is in the fluid. Increasing the resolution of the subcell mesh converges these simple numerical approximations to the

desired integrals. In this example the one-in-eight resolution on the cell interfaces shown gives at best 10 percent accuracy, while the volumes (partial areas) are only accurate to one in sixty-four, a couple of percent. A higher-resolution subcell mesh is probably desireable in this two-dimensional example.

In three dimensions, however, $10 \times 10 \times 10$ subcells resolution is usually quite accurate. The volumes are now measured to a few tenths of a percent because there are 1,000 subcells. The interface areas have 10×10 resolution and thus are accurate to a couple of percent. This is generally adequate because the grid-scale truncation error of the overall fluid algorithms is usually this large. Besides its simplicity, a potential advantage of this approach is the availability of the subcell mesh and logic for subsequent mesh refinement. In a flow solution, the partial areas and volumes are used to determine the fluxes of mass, chemical species, momentum, and energy between adjacent cells along the body surface. Cell 4 in the inset shows a cell with only a small corner in the fluid. Typical convection algorithms have problems dealing with this type of cell because of the large difference in volume between this and adjacent cells. The simplest approach to guarantee stability is to average this cell conservatively with the cell below, the cell to the right, and with cell 1. Cell 5, with only a small corner closed, presents no particular problem.

Figure 6.8 is an example of the complexity this method can handle. A Mach 5 shock is incident on a set of obstacles, the Euler equations. The geometry specification was provided by a subroutine that reads a bit map based on TEX characters and returns either a 1 or 0 depending on whether the bit at the a location was a 1 or 0. The bit map is 800×600,

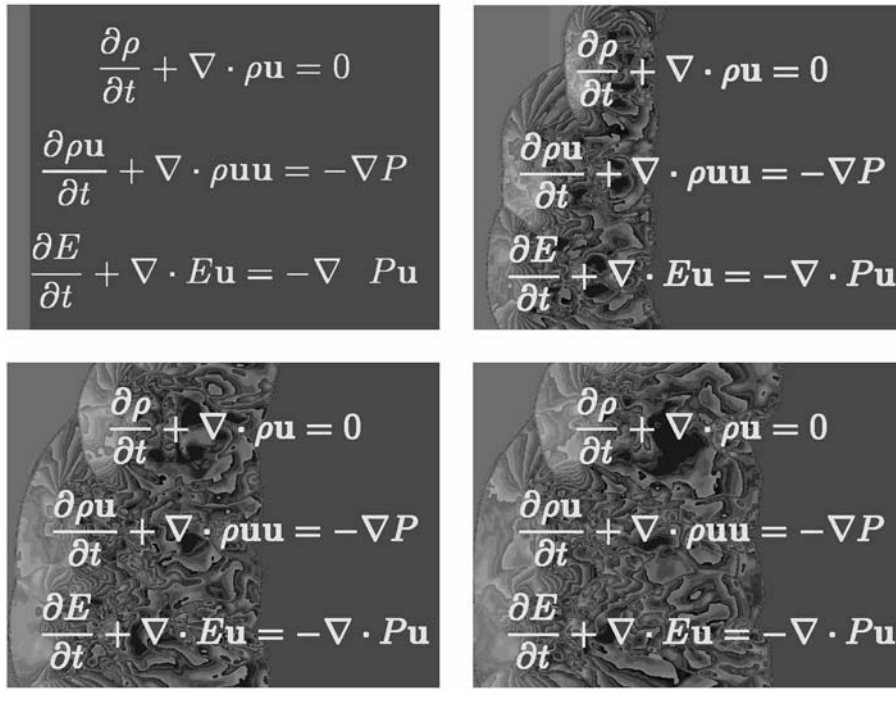

Figure 6.8. Computation of a Mach 5 shock passing through the Euler equations. Virtual cell embedding is used to treat this complicated computation on a 200×160 grid generated from a TEX bitmap.

while the grid for this calculation has 200×160 computational cells. The VCE subcell resolution is ten subdivisions for a face area and five subdivisions for the volume. The subcell resolution is adequate because the ratio of the bit map to the computational grid is approximately 4:1. The frames are taken from a movie that shows the initially planar shock reflecting off of the letters and symbols. There is no leakage of fluid into the bodies, as indicated by the fact that the ∂s and ∇s retain the initial condition inside them.

6–4. Techniques for Adaptive Gridding

Adaptive gridding techniques fall into two broad classes, *adaptive mesh redistribution* and *adaptive mesh refinement*, both contained in the acronym AMR. Techniques for adaptive mesh redistribution continuously reposition a fixed number of cells, and so they improve the resolution in particular locations of the computational domain. Techniques for adaptive mesh refinement add new cells as required and delete other cells that are no longer required. Because of the great potential of AMR for reducing computational costs without reducing the overall level of accuracy, it is a forefront area in scientific computation.

How to use a particular approach for adaptive gridding in a practical, straightforward way depends on the underlying grid and data structure. Because AMR is still in a stage of rapid development, the user should be wary of adopting any but the most straightforward techniques without first evaluating their limitations and difficulties. These decisions depend on the problems being solved, the algorithms used, and the computers on which the computational model is run. If possible, consider adopting an AMR approach that has an available library of documented utility programs for controlling the mesh data structures. The choices of the flow representation in the model and the grid generation methodology are tightly interlinked with the AMR technology. In this section, we will describe techniques for adaptive gridding in one dimension and then consider AMR in multidimensions.

6–4.1. Strategies for Refining and Coarsening a Grid

Using spatially varying resolution in localized regions of a computational grid has been common for many years in both steady-state and time-dependent calculations. For example, consider a steady-state computation of a viscous boundary layer along a flat plate. In this case, the user knows where high resolution is required. A variably spaced grid, with small cells near the plate, may be set up initially and then maintained throughout the course of the calculation. The grid spacing may be adapted to the location of boundaries, specific features of source and sink terms, and dynamic properties of the computed flow that remain relatively localized in space.

Sometimes flow regions that require high accuracy move through the grid. Then, the grid must change dynamically during the course of the calculation to keep the high resolution localized where it is needed. This is the case in models that track propagating shocks or flame fronts. There are a number of *ad hoc* criteria for placing finely gridded regions, and these are usually straightforward to program. For example, for an unsteady flame or detonation, cells may be clustered around local temperature or concentration gradients. The changes in the variables, their derivatives, and their curvatures may also be used

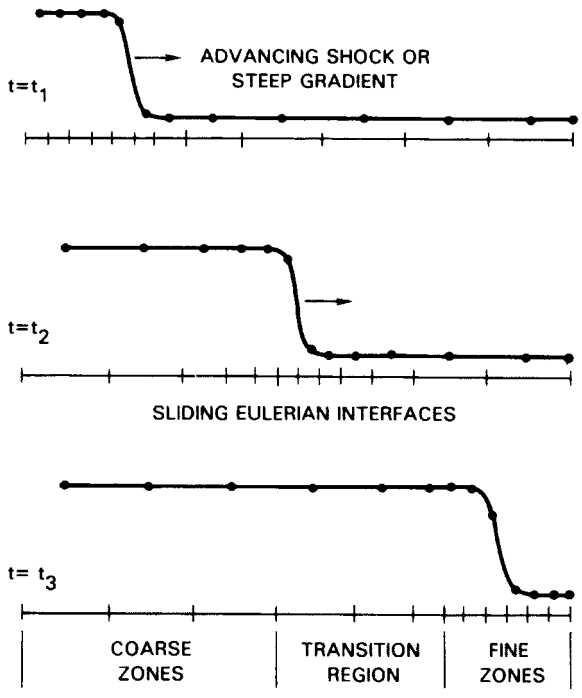

Figure 6.9. Schematic of an Eulerian sliding rezone for adaptively gridding an advancing steep gradient in a continuous, fluid profile.

to determine where resolution is needed. The cell spacing may be adjusted to minimize the overall error, and variational principles may be used to adjust the spacing. General, multidimensional formalisms have been developed for placing the adaptive grid nodes according to a variational formulation (Warsi and Thompson 1990).

The Sliding Rezone

In a *sliding rezone* method, the resolution is increased locally by sliding in cells from nearby regions where high resolution is not needed. Here both the cell interfaces and the fluid move simultaneously. Figure 6.9 illustrates a sliding rezone in one dimension. An advancing steep gradient, such as a shock or flame, moves through a region gridded with twelve cells and thirteen interfaces. The gradient is always in a region of high resolution. Uniform spacing throughout the entire domain would require twenty four cells rather than twelve, and thus would double the cost of the calculation. In many problems, zones of high resolution may be one-tenth to one-hundredth (or even less) of the size of the coarse zones. Thus clustering the cells around the gradients is an effective way to maintain high resolution with fewer cells. This approach has been particularly effective in one-dimensional and multidimensional Eulerian calculations (see for example, Oran, Young, and Boris [1978, 1982a,b]), and leads naturally to the class of multidimensional ALE methods, discussed later in this chapter.

Figure 6.9 shows a transition region between the coarse and fine zones. The size of cells in the grid should change slowly in space, perhaps only a few percent per cell. Rapid

changes in cell sizes usually reduce the accuracy of computed derivatives. Additional numerical errors occur when propagating waves are partially reflected from large cell variations. Accuracy is usually improved appreciably by gradually varying the cell sizes; the more gradual the variation in cell size, the greater the accuracy. Resolution should also be changed relatively slowly in time. As a shock moves through the grid, progressively finer cells should be introduced before the discontinuity arrives at any particular location.

One approach that ensures a relatively slow change in cell size is to use a grid-smoothing procedure, such as replacing grid locations by the average of the grid locations on the right and left. If a finely gridded region is initially ten cells wide, three or four iterations of such a smoothing algorithm may be performed before the cells at the center of the high-resolution region feel the effect. These three or four iterations are only good enough, therefore, for an overall factor of two change in cell size. If the closely spaced region is extended to twenty cells, eight or nine iterations may be used, and there are still a few uniformly spaced cells in the center. This is generally enough smoothness for an overall factor of ten change in cell size.

A caution should be issued about refining a computation within a steep gradient. This is usually quite inaccurate. Properly interpolating variables onto a grid may, in effect, require already knowing about the unresolved structure of the solution profile. In general, enough closely spaced cells must be kept around each steep gradient to maintain relatively uniform spacing in the vicinity of the gradient that is being resolved. This increases the calculation accuracy and eliminates local instabilities due to the interaction of the cell positions with steepening gradients. Programming for all these contingencies is something of an art, but a multiregion adaptive gridding procedure can give comparable results to error-minimization approaches when the minimum cell sizes are comparable. There can be a number of finely spaced regions throughout the entire computational domain, each region with its own preferred cell size. Regions of finely spaced cells can grow, shrink, split, and merge, as required by the evolving flow.

Sometimes the rate at which the cell sizes can vary is limited by the physical situation itself. A shock propagating from a region of coarse resolution to one of fine resolution develops nonphysical oscillations if the cell size changes faster than the shock can physically steepen. As a shock propagates across the coarsely gridded region using a high-resolution method, it will have a thickness of about one cell. If the shock abruptly enters a finely gridded region, this one-cell-wide profile is resolved as a ramp that spans several cells on the fine grid. Because of physical nonlinear fluid effects, this ramp profile will steepen to a one-cell shock on the finer grid. In addition, a wake of nonphysical oscillations is generated (Boris and Book 1973). These oscillations can be suppressed by changing the cell size more slowly than the shock steepens naturally. Strong shocks steepen quickly, so the cell-size transition can be rather quick. Weak shocks must travel an appreciable distance to steepen appreciably, and in these cases, the cell-size transition should be very slow. Practically, this may be prohibitively slow.

A sliding rezone also can have difficulties at material interfaces. Consider a problem containing a liquid-solid interface. As a finely gridded region comes close to the interface, cells from the solid cannot slide into the liquid without complicated logic to keep the solid and liquid from interpenetrating diffusively. This is partially circumvented by treating

some of the cell interfaces as Lagrangian to maintain sharp definition of material interfaces (DeVore et al. 1984). The remaining cells in each of the shells of distinct material move as a sliding rezone to resolve each of these distinct regions.

Splitting and Merging Cells for Grid Adaptation

Another approach to AMR is to *split* and *merge* existing cells. This also works best when these operations are performed in regions without steep gradients, where cells may be added or removed without causing numerical diffusion. The loss or gain of information is strictly localized to the place and time where the interface configuration is changed and is fully consistent with the changes in resolution desired in the representation.

For example, a single cell can be split by adding another interface anywhere between two existing interfaces, and then setting the values of the variables in the two new cells equal to those in the old cell. This does not improve the resolution of steep gradients because no intermediate, averaged values are generated. This also leaves terraces composed of two adjacent cells with equal values.

A better composite regridding operation composed of two splittings and a merging is shown in Figure 6.10a,b,c. Fractions of adjacent cells are split off and then merged together to form a smaller cell of intermediate value. Merging, shown in Figure 6.10d,e,f, decreases the resolution and speeds up the calculation by reducing the number of cells and by allowing a larger timestep. It also has an effect similar to numerical diffusion.

It is possible to program criteria that can be used to split and merge cells automatically and in a way that adapts to the resolution needed in the evolving flow. Because these processes result in discontinuous changes in the number of cells and interfaces in the computational domain, the bookkeeping is somewhat more complicated than when the number of cells is constant. This drawback is also a concern when attempting to construct a model that is load balanced for parallel processing. Both the number of grid points and the location in computer memory of data referring to a grid-point change as cells are subdivided or removed. In one dimension, this added complexity is not severe, and even in two and three dimensions this complexity is usually worthwhile.

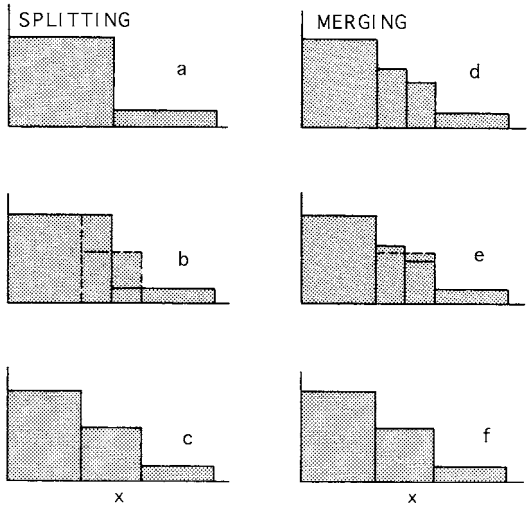

Figure 6.10. Schematic showing a discontinuous Lagrangian rezoning procedure. Splitting: a, b, and finally c. Merging: d, e, and finally f.

In flame or shock calculations, the grid must be refined *before* a gradient enters the cell or else numerical diffusion is added by the interpolation process. The conditions chosen for splitting cells ahead of an advancing gradient, and merging them after the gradient passes, determine the accuracy and cost of the overall algorithm. If the splitting is not performed far enough ahead of the approaching gradients, some of the region needing fine resolution could move into a region where the cells are too large. If the splitting is done too far ahead of the gradient, or over too large a region, a large amount of computing may be wasted. When the merging occurs too soon after the transient has passed, sharp gradients diffuse faster than physical transport would allow. When the merging occurs too late, the advancing steep gradient region leaves more and more fine cells behind and the calculation slowly grinds to a halt. We recommend allowing only one splitting or merging per timestep at a given location.

The procedure for AMR is more complicated in the Lagrangian case than in the Eulerian case. In addition to the grid redistribution in the AMR, there is grid redistribution provided naturally by the Lagrangian framework itself through the concentration of grid points in regions where the fluid is compressing. Nonetheless, there are important situations, such as a flame front, where fine resolution is required but the fluid is actually expanding. The practical problem is how to arbitrarily concentrate grid points in a Lagrangian calculation and still maintain the desirable property of no numerical diffusion. The mesh redistribution procedures used in Eulerian calculations are numerically diffusive whenever the grid moves relative to the fluid. Therefore, Lagrangian representations and the associated flow algorithms would generally seem to be better suited to the splitting and merging strategy than to sliding-rezone approaches. For multidimensional Lagrangian models, as discussed below, it is useful to have some form of ALE capability, which is a type of sliding rezone, in addition to cell splitting and merging.

6–4.2. Adaptive Mesh Redistribution on Multidimensional Cartesian Grids

Advantages of using Cartesian grids were mentioned in Section 6–3.4. Another advantage associated with moving or variably spaced Cartesian grids is that relatively few additional computer operations are required. Thus the models can be highly optimized. For these reasons, and despite some shortcomings, rectilinear grids are attractive for complex flows and geometries.

The main one-dimensional AMR techniques described in Section 6–4.1 for the one-dimensional case can be extended to multidimensions when orthogonal, rectilinear grids are used. A two-dimensional axisymmetric application of this approach is shown in Figure 6.11 (Lind et al. 1997, 1998). The high-resolution region of the grid has 4 cm $\times$ 4 cm zones near the surface of the reaction vessel. This stretches to cells of 50 cm $\times$ 100 cm at the outer boundaries of the computational domain, putting the outer boundary far away without affecting the resolution on parts of the grid that need the finer resolution. AMR is done in each direction independently without affecting the location of the grid lines in the other direction. Without the mesh redistribution, a factor of 25 more cells would be required to have equivalent resolution in and around the chamber.

Figure 6.12 shows a three-dimensional example of an even more complex geometry using a stretched Cartesian mesh to represent the above-water superstructure of a naval

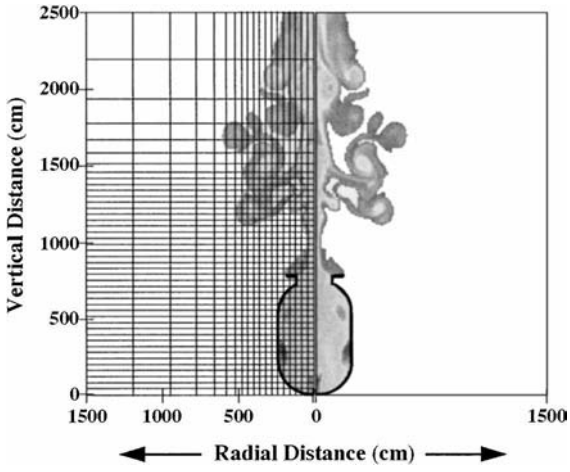

Figure 6.11. An illustration of two-dimensional Cartesian AMR, applied to the complex geometry of a detonation incinerator (Lind et al. 1997). The figure shows the central portion of the axisymmetric domain, where the axis of symmetry is vertical through the center of the figure. The gray scale maps the instantaneous oxygen density at 150 ms after a detonation of fifty pounds of explosive. The left half of the figure is overlaid with the stretched Eulerian grid, for which every tenth line of the 250×500 $r - z$ grid is shown.

vessel (Landsberg et al. 1993; Landsberg and Boris 1997). The grid was generated using VCE (Section 6–3.4), and has $320 \times 98 \times 128 = 3{,}932{,}160$ cells. The resolution near the smokestacks is uniform, consisting of cubes that are approximately 0.3 m on a side. The grid then stretches to fill a 760 m $\times$ 90 m $\times$ 210 m computational domain. If the gridding were uniform at the smallest cell size, 5.25×10^8 cells would have been used. Thus simply stretching the grid, as illustrated here, has reduced the cost of this three-dimensional computation of the hot stack gases by more than a factor of 130.

Another useful application of Cartesian mesh redistribution is to smooth small, grid-induced irregularities that appear, for example, as unphysical ripples near shocks or slip lines that are oblique to the grid. These ripples often occur with low-dissipation, high-resolution solution algorithms that can maintain a discontinuity within one or two cells. A useful computational "fix" for these ripples is to move the grid lines forward a small amount at one timestep, and then backward at the next timestep, effectively oscillating the grid. Each back and forth movement should be about 25 percent of the local cell

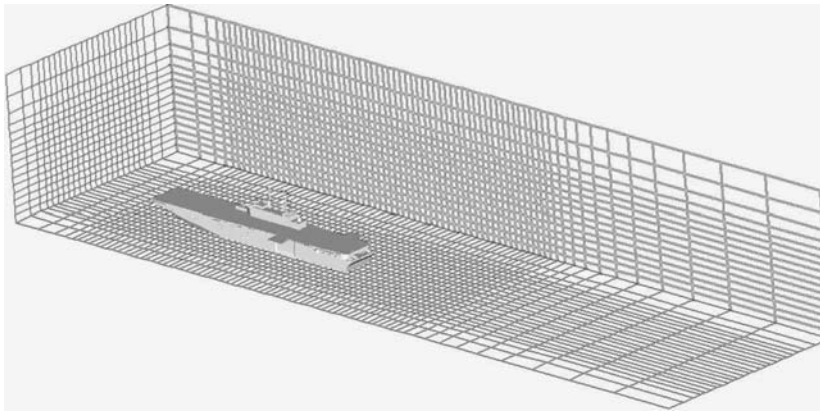

Figure 6.12. Three-dimensional example of a stretched Eulerian grid with very complex ship superstructure geometry. This is part of a simulation used to study the dispersal of gases from the ship's stacks (Landsberg and Boris 1997).

spacing. This oscillatory technique appreciably reduces local fluctuations in the solution arising from a discrete grid. It requires algorithms that allow the grid to be changed from one timestep to the next, which is generally not a difficult extension for many Cartesian solution algorithms.

In two or three dimensions, as in the case of one dimension discussed above, a rapid variation of cell size often results in large errors. The size of these errors depends on how fast the solution changes on the grid and on the particular algorithm used. In adaptive refinement, as described below, there are may be factors of two or greater in the sizes of neighboring cells. This large variation in cell size should not occur where the function is changing rapidly. Slow, smooth changes will minimize the inevitable errors. This, in effect, says that if the grid changes are large (factors or two), more fine resolution is required to ensure that the function does not change much from cell to cell. Algorithms for controlling errors at such grid interfaces help to reduce the amount of fine grid needed to reach smooth regions of the flow away from steep gradients (see, for example, Edwards [1996], Kravchenko, Moin, and Moser [1996]). The trade-off is between additional algorithm complexity and an increased number of cells.

More difficult problems arise where there are moving bodies in a stationary Cartesian grid. One method for treating this is discussed in Section 6–3.1. For the vessel shown in Figure 6.11, flapper plates that trap hot reaction products in the vessel after the explosion. If the grid changes in time, recomputing the partially obstructed cell geometry factors must be done at every timestep rather than just once at the beginning of the computation. The techniques for doing this, Dynamic Virtual Cell Embedding (DVCE), are currently under development (Emery et al. 1997).

One shortcoming of a multidimensional sliding rezone in a global Cartesian grid is illustrated in Figures 6.11 and 6.12. A region of fine cells extends all the way across the domain in some directions, whether or not this resolution is needed out to the boundaries. This means that resolving complicated structures in one part of the grid requires keeping closely spaced cells in other regions where they may be of little use. It also means that there are cells with rather large aspect ratios, and the solution algorithms must accomodate this.

There are, however, many problems in which having these added cells is not prohibitive. The gains in simplicity and local accuracy often outweigh the cost of additional cells. In the example in Figure 6.11, about half of the cells are in the finely gridded region extending out to 6 m in radius and 16 m in height. Thus the cost in additional cells to represent the outer 98 percent of the domain is only about a factor of two overall even though the finely gridded regions extend right to the boundaries in both the r and z directions. The advantage of Cartesian grids is that they are straightforward to parallelize and so produce scalable code that can take advantage of the highest-performance computers.

6–4.3. Adaptive Mesh Refinement on Multidimensional Cartesian Grids

Figure 6.13 may be used to compare several different ways to adaptively refine a computational grid when the single, uniform Cartesian grid, shown in Figure 6.13a, does not give adequate resolution. The curved line in each of the six panels represents a steep front advancing towards the lower right corner of the figure. For example, the front could be a flame and the shaded region would then be the combustion products. Better resolution is

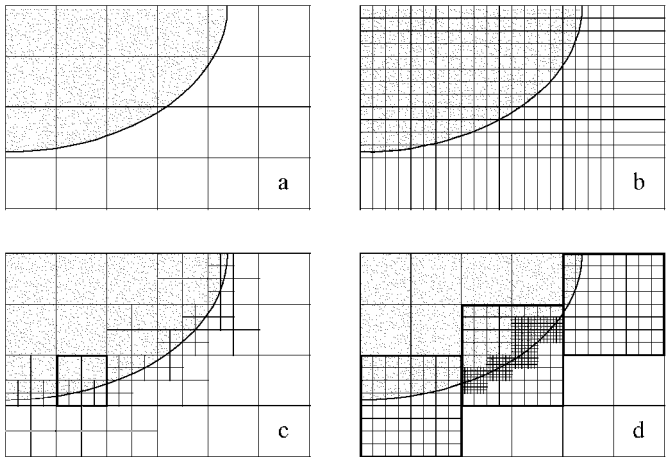

Figure 6.13. Several AMR grids for representing a steep front: (a) uniform resolution; (b) adaptive mesh redistribution; (c) single-cell refinement; (d) block refinement.

needed to compute accurately in the flame than is required in the regions on both sides of the advancing front. Figure 6.13a shows uniform but poor resolution with 24 cells. Figure 6.13b shows how adjusting the spacing of the grid planes in the Cartesian grid has not been very helpful. There is a factor of four improvement in resolution using AMR, but it requires 273 cells compared to 24 cells with coarsely, uniform gridding. For this problem, adaptively spacing the Cartesian grid lines brings virtually no gain because the fine cells must extend to the boundaries of the domain.

Figure 6.13c shows one strategy for Cartesian AMR in which cells with steep gradients or cells adjacent to the body are subdivided by a factor of two in each direction (Young et al. 1991). The cells are divided in two in each direction, although this could have been any refinement factor, and need not even be the same factor in each direction. Two levels of refinement are shown in Figure 6.13c. Some of the cells divided by two in both X and Y are again subdivided to give the fourfold resolution improvement of the front shown in Figure 6.13b. Here 101 cells would give just as good a solution as the 273 cells of Figure 6.13b. This is the type of improvement obtained, for example, by Khokhlov (1998) using the fully threaded tree for the data structure. The four cells in the lower left in Figure 6.13c is one of the original single cells subdivided in the first level of refinement. Figure 6.14 illustrates an application of this technique to the computation of a deflagration-to-detonation transition in the region of a flame brush (Khokhlov and Oran 1999). In this application, a full set of reactive Navier-Stokes equations are solved on a mesh that adapts as the flow changes.

Figure 6.13d shows another AMR approach in which patches of cells, taken as a block, are all subdivided. This is the approach of Berger and Oliger (1984), Berger and Colella (1989), and Bell et al. (1994). In general, the patches can replace any number of cells. In the example in Figure 6.13d, 2×2 patches of background cells are replaced by 8×8 patches, a fourfold improvement in resolution in both directions. Here there are two parameters free: the size of the patches, which need not all be the same or even be square; and the resolution refinement factor in each patch, which need not be a factor of two. In

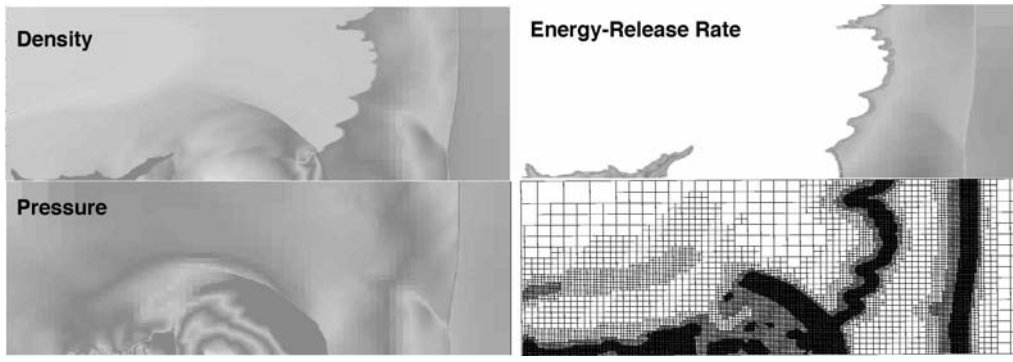

Figure 6.14. Selected physical quantities and the computational grid from one timestep of a computation of detonation-to-detonation transition in a mixture of acetylene and air. The refined grid tracks steep variables in the flow: the leading shock on the far right, the turbulent flame (convoluted surface to the left of the shock), and the explosion and burning region on the bottom left. AMR based on the fully threaded tree data structure (Khokhlov 1998; Khokhlov and Oran 1999).

Figure 6.13d, a second level of refinement is shown in the central patch, where five higher resolution patches have been injected to increase the resolution by a factor of sixteen near the flame. Using 2×2 patches, which means a fourfold resolution increase in each patch, requires 204 cells at the first level of refinement to obtain a solution comparable to that obtainable with grids in Figure 6.13b or c.

Figure 6.15 is an example of AMR on a Cartesian grid used for global atmospheric modeling of the stratosphere (Nikiforakis and Toro n.d.; Nikiforakis et al. n.d.). The AMR technique, which is similar to that developed by Berger and colleagues, produces mesh of the general type in Figure 6.13d. The figures on the left show computed contours of an advected scalar (that might represent temperature or a species density, such as ozone) at an altitude of about 18 km superimposed on a background map of the earth. The map projection looks down on the earth from above the North Pole, which is at the center of the figure, and extends to the equator at the edges. The figures on the right show the AMR grid used for the computation. An important element in stratospheric ozone depletion is the movement of the filamentary and tubular structures such as those seen in the figures.

There could be advantages to refining the grid with larger patches of cells than the limiting 1×1 patch shown in Figure 6.13c, depending on the underlying data structure. Sometimes using larger patches significantly decreases the amount of computer storage and number of logical operations associated with defining the grid connectivity and resolution. Such extended patches, however, can require considerably more cells and computer time for floating-point computations because the ability to localize grid resolution changes is somewhat limited.

In one way of looking at the Cartesian AMR, these variations are not really different methods, just different choices of parameters governing the extent of the patches and the refinement factor at each level. There could even be a more complex strategy in which these parameters are free and optimized locally, depending on what is being resolved at different places. Such complexity, however, usually brings little advantage and makes the program harder to write and run efficiently. In fact, how to choose these controlling parameters is very much affected by the strategy adopted for parallel processing. If the individual grid

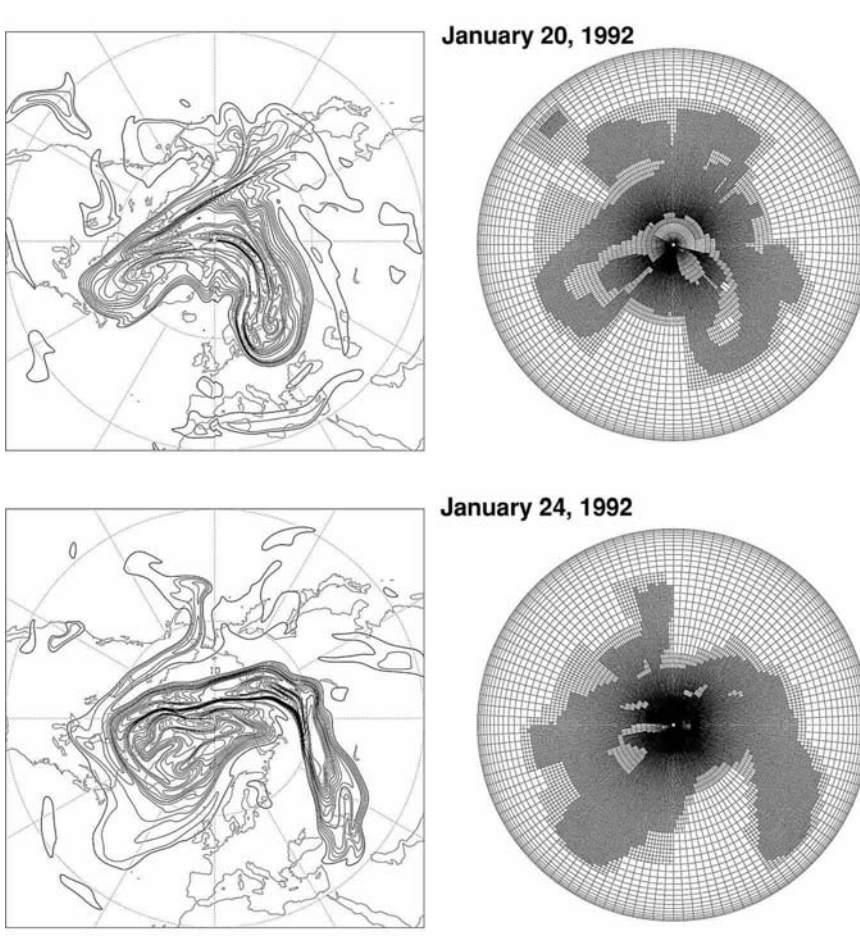

Figure 6.15. AMR on a Cartesian grid used to model the stratosphere (Nikiforakis and Toro 1999; Nikiforakis et al. 1999). Results are shown at typical times on two days in January 1992. The figures on the left show computed contours of an advected scalar (which might represent temperature or a species density, such as ozone) at an altitude of about 18 km superimposed on a background map of the earth. The projection looks down from above the North Pole, which is at the center of the figure, and extends to the equator at the edges. The figures on the right show the AMR grid used for the computation. The coarsest grid has 120×60 cells, giving a coarse resolution of 3×3 degrees, followed by two levels of refinement, giving a maximum resolution of 0.5×0.5 degrees. Typical timesteps for such a computation are on the order of a few minutes. (Courtesy of N. Nikiforakis)

patches are large enough, and if the logic for connecting patch solutions at their boundaries to the grid in which they are embedded is simple enough, each patch can be executed on a separate processor. This greatly simplifies load balancing on multiprocessor computers.

6–4.4. AMR and Overset Grids

Because the basic structures of the overset grid approach are the separate grid patches, the extensions to AMR are straightforward conceptually. Away from the body surfaces, the patches can be Cartesian and adapt to flow features without the complication of being

body fitted. The patches can be nested for multilevel mesh refinement (Kao, Liou, and Chow 1994) or moved to redistribute the mesh (Meakin 1995). The key to AMR for overset grids is a fast automatic program to recompute the interpolation factors and to keep track of the hierarchy of grids in overlapping regions. When the composite grid is static or moves in a prescribed way, these quantities may be precomputed. In the general case, these bookkeeping grid computations must be performed efficiently without programmer intervention. In section 6–4.5, we consider embedded and multigrid AMR as a way to simplify and speed up the general overset approach.

In many applications of overset grids, all of the cells are advanced with the same timestep determined by the smallest grid spacing. There are several ways of relaxing this timestep restriction. In multiple-grid Cartesian AMR, the finely spaced regions are often integrated with smaller timesteps. The boundary conditions used are the solutions at the borders between the fine and coarse grids (Berger and Jameson 1985; Thompson et al. 1985). One major computational problem centers around matching the different grids at their interfaces. Another problem is maintaining conservation of the variables, which is usually difficult to ensure throughout a changing grid. A third problem is determining criteria for when to advance different regions of the grid.

Examples of applications of this approach are shown Figure 6.16, which shows gridding for an integrated space shuttle vehicle (Buning et al. 1998). Here each component of the configuration is gridded separately and overset onto a main grid to form the complete grid. The main grid consists of the external tanks. Minor grids are generated around the rest of the bodies and other special features of the flow. The interconnection routines identify when points of one grid fall within a body boundary of another (these are called grid-hole points) and give the interconnection data so one grid can provide boundary data to another. In this figure, each grid component was generated independently. Figures 6.16b,c,d show constant planes from each of the main grids in a region near the trailing edge of the orbiter.

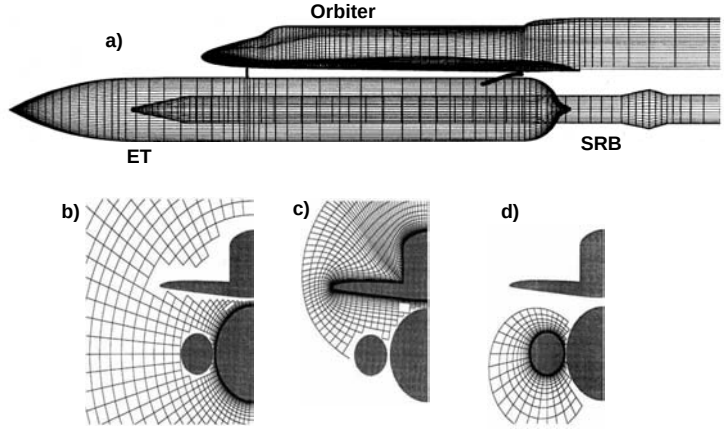

Figure 6.16. Overset Chimera gridding for integrated space shuttle vehicle. (a) Surface grid of the integrated vehicle: the orbiter, external tank (ET), and the one of the solid rocket boosters (SRB) and the attached hardware. (Courtesy of R. Meakin) (Buning et al. 1998) Sections through the integrated vehicle: (b) external tank grid; (c) orbiter grid; (d) SRB grid.

6–4.5. AMR and Unstructured Grids

Unstructured grids are used with Eulerian, Lagrangian, and ALE algorithms. Thus adaptive mesh redistribution and adaptive mesh refinement are both needed, depending on the model and the application. The rapid distortion in a Lagrangian or ALE grid in multidimensional flow can be handled by using a grid of triangles in two dimensions and tetrahedra in three dimensions. Reconnecting the grid nodes to ensure compact, roughly equilateral triangles is naturally accomplished with unstructured grids. We saw in Section 6–3.3 that unstructured grids permit a natural representation of complicated shapes and have been applied extensively to Lagrangian representations of the flow equations. When the unstructured representation is applied to Lagrangian fluid problems, the grid was adaptively reconnected (Crowley 1971; Fritts and Boris 1979; Crowley et al. 1985).

There was a considerable effort to develop fully Lagrangian flow algorithms on unstructured grids. The primary impediment to achieving this in multidimensions is that the rotational components of the flow quickly distort the Lagrangian grid to the point where the calculation becomes meaningless. Few problems using multidimensional Lagrangian grids can be treated satisfactorily without some form of adaptive gridding that allows fluid to slide across the grid, thus making the overall model an ALE representation. A common approach is to apply an interpolative regridding procedure every few cycles. With this regridding, the composite effect is usually as diffusive as starting with an Eulerian algorithm.

Current adaptive unstructured-grid representations are often programmed using ALE grid strategies. These take advantage of the flexibility to adjust the grid without the main difficulties of dealing with completely Lagrangian grid motion. Grid adaptation is usually controlled by heuristic formulas based on criteria using the fluid dynamic derivatives such as gradients and second derivatives of pressure and density. The strategy is to distort the existing grid without remeshing, but to correct severe distortions by actually changing the topology of the grid. Whenever greater resolution is needed, additional grid nodes are added just ahead of the region requiring fine resolution, and the grid is coarsened by removing cells where the spatial gradients are diminishing.

Chapter 8 describes finite-element (FE) techniques for constructing solution algorithms. The FE formalism uses nonlocal solution algorithms that lead to the inversion of sparse matrices. This means that FE techniques are more expensive than the standard finite-difference or finite-volume methods. The added cost of using irregular, nonorthogonal, and even unstructured grids is a relatively small addition to the cost of the basic solution method. Further, because of the nonlocal nature of FE methods, local grid problems (such as rapidly changing or distorted cells) are less important. Small cells that arise as the solution evolves can be treated by moving the grid nodes without changing the topology of the grid connectivity.

Two examples of unstructured, finite-element grids applied to complex geometries are shown in Figures 6.17 and 6.18. Figure 6.17 is a supersonic rocket exhaust through a nozzle, which was computed using the explicit finite-element FCT (FEMFCT) approach (Ramamurti, Kailasanath, and Löhner 1992). Figure 6.18 is part of an implicit incompressible computation of the thrust a tuna generates on its body by oscillating its caudal fin

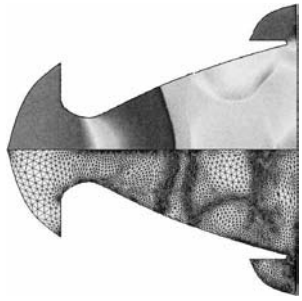

Figure 6.17. AMR of the unsteady shock structure in a rocket nozzle using h-refinement on an unstructured-grid. Upper half: pressure field showing shocks and rarefactions. Lower-half: instantaneous computational grid. (Ramamurti et al. 1992)

(Ramamurti, Sandberg, and Löhner 1999). FE methods are generally used for problems with complex flows in complex geometries.

The FE methodology may be extended to solve additional equations that determine the optimal locations of the grid nodes. In general, such adaptive gridding based on a variational principle is complicated and expensive on arbitrarily complex multidimensional grids. Because the auxiliary calculations associated with adapting the grid in this way can become as expensive as integrating the physical variables, it is generally not used.

H-refinement or H-enrichment (see, for example, Löhner [1987]; Löhner [1989]) is a more popular and effective method for finding a better grid than solving the additional equations for the locations of grid nodes. In this approach, the grid is generally held fixed in space and new tetrahedra are formed by subdividing individual triangles and tetrahedra. The process is a multidimensional generalization of the splitting and merging processes described in Section 6–4.1. A tetrahedron may be subdivided into two, four, or eight smaller terathedra (Löhner and Baum 1990). This is programmed to occur automatically in response to a passing shock, flame, vortex, or contact surface. Automatic coarsening of the grid in regions where the solution gradients are decreasing is also necessary for overall efficiency. A nondimensional error criterion based on the ratio of second derivatives to first derivatives on the grid incorporates all of the necessary criteria. This approach has

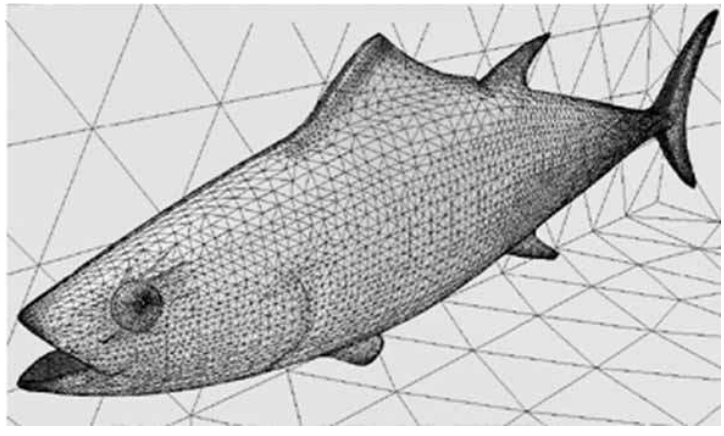

Figure 6.18. Frame taken from a finite-element computation of a tuna moving through water. The unstructured grid of small tetrahedra near the tuna stretches as it moves away from the tuna. Dark regions on the image indicate high pressure. (Ramamurti et al. 1999)

proven to be quite successful for a wide range of problems but has not yet been applied successfully to production codes on parallel processors where load-balancing is important.

6–5. Resolution and Accuracy on a Grid

Quantitative measures of the resolution of a representation and the accuracy of an algorithm are usually based on its applications to idealized test problems. The convection of a structure at constant velocity (linear convection) on a uniformly spaced grid was the test problem used in Chapter 4. Such comparisons to a known solution to the equations are a useful guide to *verify* that the numerical model actually implements the equation set correctly and that the numerical algorithms used are properly programmed. These idealized verification tests are difficult to define precisely when you are testing programs and algorithms for irregular gridding, complex geometry, and complex flow patterns. Further, they do not address the question of whether the equations implemented accurately represents the correct physics and chemistry.

Because of these limitations, the fluid-dynamic and reactive-flow simulation communities generally recognize a second class of tests called *validations* or *calibrations*. Here the results of the numerical model are *calibrated* against known data sets from experiment or from more fundamental detailed simulations that themselves have undergone careful scrutiny for fidelity and accuracy. A body of work is beginning to build up around attempts to "quantify the uncertainty" in fluid-dynamics and reactive-flow simulations (Roache 1997). This subject, however, is contentious and is subject to a dose of semantic games that parade as rigor. Sometimes the deliberations are politically motivated, and resulting evaluations are used for personal and political purposes rather than being true scientific tests.

What is important here is the realization that we cannot expect to know how accurate a computation is when the problem being solved is truly new. Furthermore, these new problems are the ones of greatest value. Only a lucky few can make a living by solving problems for which everyone already knows the answer. Therefore, it is valuable to optimize the overall properties and accuracy of the representation and algorithms in a model. For example, the ability to ensure the global physical properties of conservation, positivity, and causality on both uniform and complex grids provide consistent, meaningful ways to control error. Because these properties are expected to hold generally, they can still be tested quantitatively in complex situations, even when the solutions to the problem are not known in detail.

6–5.1. Errors Introduced by Grid Representations

Table 4.4 lists four types of errors that occur in numerical simulations: local errors, amplitude errors, phase errors, and Gibbs errors. These last three "nonlocal" errors arise from the representation, the discretization, or the algorithm chosen to represent the convective derivative. Indeed, in discussing errors, it is generally impractical to separate the representation from the algorithms implemented in that representation. Nevertheless, it is useful to look again at these types of errors in light of the additional information presented in this chapter.

Phase errors are a nonphysical form of dispersion in which different wavelength harmonics are numerically convected at different speeds. They adversely affect causality, positivity, and accuracy. They are of particular concern in algorithms for convection, because they continually accumulate and may not reach a limiting level. Amplitude errors occur when an algorithm is either unstable or excessively damped numerically. Amplitude errors tend to be self-limiting when the overall algorithm is strictly conservative and positivity preserving. If mode amplitudes grow nonphysically, as explained earlier, the algorithm is said to be unstable. A positive fluctuation in density, for example, cannot grow without bound using a positivity-preserving conservative algorithm because the density will have to become negative somewhere else to ensure conservation, and this cannot happen. On the other hand, if the variation in a particular mode is decreasing it can only decrease to the point at which the density is constant.

The Gibbs phenomenon, or Gibbs errors, that occur in the solution of wave and convection equations, look very much like phase errors. They are *not* due to the algorithm but rather to the discrete nature of the representation itself. Thus they cannot be eliminated by improving the phase and amplitude properties of the numerical algorithm and cannot be reduced by the quality of the numerical grid. Gibbs errors originate in the missing short-wavelength harmonics, shorter than two computational cells. These are not included in the representation. They constitute a kind of uncertainty with the result that the structure of the continuous solutions within each cell cannot be resolved.

These unresolved spatial variations are an inherent error associated with representing a continuous but steep function on a discrete grid. In numerical simulations, Gibbs errors appear as undershoots and overshoots that are larger near steep gradients and discontinuities and become apparent as soon as a sharp profile moves a fraction of a cell from its initial location. Gibbs errors set a bound on the accuracy possible in discrete approximations. Such errors can be reduced only by improving the resolution in an Eulerian representation. By moving the nodes in a Lagrangian manner, Gibbs errors are only partially suppressed, and this occurs at the cost of additional degrees of freedom and more expensive algorithms.

Phase, amplitude, and Gibbs errors make it very difficult to represent and move sharp gradients or small-scale, localized flow structures with algorithms based on highly nonlocal expansions. This is particularly true on complex, irregular grids. The basic problem is that the shortest wavelengths are responsible for the steepest portions of the profile, and are also subject to the greatest phase and amplitude errors. To use a nonlocal expansion requires using many basis functions and ensuring that their contributions are all properly phased so that they add up to a steep transition with smooth regions on either side. While this proper phasing is possible on regular and smoothly varying grids, if the phases and amplitudes of any of the superimposed harmonics are not maintained properly, as occurs on irregular grids, the solution deteriorates quickly.

Therefore, in representing a discontinuity numerically, a balance must be established between the number of degrees of freedom that are used to represent the discontinuity and the accuracy with which each degree of freedom is integrated. We have found that best overall results are obtained by methods of intermediate order, with fairly local expansion functions, and with up to fourth- but not much higher-order phase errors. Second-order amplitude errors are often acceptable, but phase errors accumulate in time and so must be kept small.

6–5.2. The Influence of Local Errors

Discretizing spatial derivatives usually generates the largest errors in reactive flows. Thus local errors, which include round-off, truncation, and lack of conservation in numerical solutions of local processes, as discussed in Chapter 5, are not directly relevant to the errors induced by gridding and spatial resolution. Local errors, however, are important indirectly through their coupling to spatial variations. For example, local errors become important when inaccuracies in solving chemical-reaction equations disturb the overall energy flow in a system. Another way local errors can be important is through small local round-off or truncation errors in density that appear as random waves in the pressure and eventually show up in the velocity. If the material is nearly incompressible, these errors resemble the continuous generation of fairly large amplitude sound waves. Local errors can also generate apparent gradients that may trigger adaptive mesh refinement, increasing the cost of a calculation through the additional cells generated to resolve the spurious gradients.

Often derivatives are approximated, either directly or indirectly, as differences between the values in neighboring cells. When these values are nearly equal, round-off errors are enhanced because the difference is determined by the last few digits of the adjacent values. To control this, as a rule of thumb, we would like to keep round-off errors appreciably smaller, say several orders of magnitude smaller, than the truncation errors in the numerical algorithms. This requirement translates into a need for about seven decimal digits of accuracy in the real numbers calculated. Practically, this means that 32-bit, floating-point numbers are usually adequate to suppress round-off effects. Nevertheless, always be aware of round-off because there are situations, particularly in Lagrangian representations and in incompressible flows, in which even 64-bit double-precision accuracy may not be enough.

6–5.3. The Connection between Resolution and Accuracy

The piecewise-constant and piecewise-linear representations of Figure 6.2 result from superimposing relatively well-localized expansion functions. Both of these representations use the same grid. The number of degrees of freedom are almost the same and the resolution is virtually identical. Nevertheless, the piecewise linear discretization seems to be a better approximation because the approximating function is continuous everywhere, even though its derivatives are not. By using the same resolution, but interpreting the representation differently, different accuracy approximations are possible. This illustrates the inherent trade-off that must be made between the number of degrees of freedom in a representation, typically the resolution, and the amount of work done to advance each degree of freedom.

In general, higher-order methods are better than lower-order methods. There is, however, a point of diminishing returns at which the extra work of computing the higher-order solution and keeping the method stable does not really bring accompanying increases in accuracy. The higher-order methods can only yield better answers for scale lengths that are longer than a few cells. Interpolation, no matter how high the order, also cannot reproduce unresolved variations occurring within a computational cell. This is a fundamental

limitation set by the resolution of the representation. It is independent of the order of accuracy, the type of expansion, or the algorithms used. The Gibbs errors discussed earlier arise from this fundamental computational limitation.

The representation of time and space variations may be optimized in different ways depending on the particular problem. Whether point values, volume-averaged values, or Fourier coefficients are given, the goal is a good representation of a particular function with a minimum number of degrees of freedom. When the real function is smooth with the exception of a few sharp steps, the optimal representation is very different from the case in which the function has an oscillatory structure spread over many spatial scales.

REFERENCES

Aftosmis, M.J., J.E. Melton, and M.J. Berger. 1995. Adaptation and surface modeling of Cartesian mesh methods, AIAA paper 95-1725-CP, Reston, VA: American Institute of Aeronautics and Astronautics.

Aftosmis, M.J., M.J. Berger, and J.E. Melton. 1997. Robust and efficient Cartesian mesh generation for component-based geometry, AIAA 97-0196, Reston, VA: American Institute of Aeronautics and Astronautics.

Amsden, A.A., H.M. Ruppel, and C.W. Hirt. 1980. SALE: A simplified ALE computer program for fluid flow at all speeds, Los Alamos National Laboratory report LA-8095, June 1980, Los Alamos, NM.

Anderson, J.D., Jr. 1995. *Computational fluid dynamics*. New York: McGraw-Hill.

Arakawa, A. 1966. Computational design for long-term numerical integration of the equations of fluid motion: Two dimensional incompressible flow. Part I. *Journal of Computational Physics* 1:119–143 (reprinted in the *Journal of Computational Physics* 135:103–114, 1997).

Bell, J., M. Berger, J. Saltzman, and M. Welcome. 1994. Three-dimensional adaptive mesh refinement for hyperbolic conservation laws. *SIAM J. Sci. Comput.* 15:127–138.

Benedetto, J.J., M.W. Frazier, eds. 1994. *Wavelets – Mathematics and applications*. Boca Raton, Fla.: CRC Press, Inc.

Benek, J.A., P.G. Buning, and J.L. Steger. 1985. A 3D Chimera grid embedding technique. AIAA paper 85-1523, presented at AIAA 7th Computational Fluid Dynamics Conference, Cincinnati.

Benek, J.A., T.L. Donegan, and N.E. Suhs. 1987. Extended Chimera grid embedding scheme with applications to viscous flows. AIAA paper 87-1126–CP, presented at AIAA 8th Computational Fluid Dynamics Conference, Honolulu.

Benek, J.A., J.L. Steger, and F.C. Dougherty. 1983. A flexible grid embedding technique with application to the Euler equations. AIAA paper 83-1944, presented at AIAA 6th Computational Fluid Dynamics Conference, Danvers, Mass.

Benson, D.J. 1992. Vectorization techniques for explicit arbitrary Lagrangian-Eulerian calculations. *Computer Methods in Applied Mechanics and Engineering* 96:303–328.

Berger, M.J., and P. Colella. 1989. Local adaptive mesh refinement for shock hydrodynamics. *Journal of Computational Physics* 82:64–84.

Berger, M.J., and A. Jameson. 1985. An adaptive multigrid method for the Euler equations. In *Ninth International Conference on Numerical Methods in Fluid Dynamics*, eds. Soubbaramayer and J.P. Boujot, 92–97. New York: Springer-Verlag.

Berger, M.J., and R.L. LeVeque. 1989. An adaptive Cartesian mesh algorithm for the Euler equations in arbitrary geometries. AIAA paper 89-1030-CP, Reston, VA: American Institute of Aeronautics and Astronautics.

Berger, M.J., and J.E. Melton. 1994. An accuracy test of a Cartesian grid method for steady flow in complex geometries. In *Proceedings of the 5th International Conference on Hyperbolic Problems*.

Berger, M.J., and J. Oliger. 1984. Adaptive mesh refinement for hyperbolic partial differential equations. *Journal of Computational Physics* 53:484–512.

Beylkin, G., and J.M. Keiser. 1997. On the adaptive numerical solution of nonlinear partial differential equations in wavelet bases. *Journal of Computational Physics* 132:233–259.

Bird, G.A. 1994. *Molecular gas dynamics and the direct simulation of gas flows*. Oxford, England: Clarendon Press.

Boon, J.P. 1991. Statistical mechanics and hydrodynamics of lattice gas automata. *Physica D* 47:3–8.

Boris, J.P. 1986. A vectorized "near neighbors" algorithm of order *N* using a monotonic logical grid. *Journal of Computational Physics* 66:1–20.

Boris, J.P., and D.L. Book. 1973. Flux-corrected transport I: SHASTA – A fluid transport algorithm that works. *Journal of Computational Physics* 11:38–69.

Brackbill, J.U. 1976. Numerical magnetohydrodynamics for high-beta plasmas. *Methods in Computational Physics* 16:1–41.

Brackbill, J.U., and H.M. Ruppel. 1986. FLIP – A method for adaptively zoned, particle-in-cell calculations of fluid flows in two dimensions. *Journal of Computational Physics* 65:314–343.

Buning, P., I. Chiu, S. Obayashi, Y. Rizk, and J. Steger. 1998. Numerical simulation of the integrated space shuttle vehicle in ascent. AIAA-88-4359-CP, presented at AIAA Atmospheric Flight Mechanics Conference, Aug. 1998, Reston, VA.

Canuto, C., M.Y. Hussaini, A. Quarternoni, and T.A. Zang. 1988. *Spectral methods in fluid dynamics*. Berlin: Springer-Verlag.

Castillo, J.E. 1991. *Mathematical aspects of numerical grid generation*. Philadelphia: Society for Industrial and Applied Mathematics.

Chen, S., and G.D. Doolan. 1998. Lattice Boltzmann method for fluid flows. *Ann. Rev. Fluid Mech.* 30:329–364.

Chesshire, G., and W.D. Henshaw. 1994. A scheme for conservative interpolation in overlapping grids. *SIAM J. Sci. Comput.* 15(4):819–845.

Crowley, W.P. 1971. FLAG: A free-Lagrange method for numerically simulating hydrodynamic flows in two dimensions. In *Proceedings of the Second International Conference on Numerical Methods in Fluid Dynamics*. ed. M. Holt, 37–43. New York: Springer-Verlag.

Crowley, W.P., M.J. Fritts, and H. Trease, eds. 1985. *The free Lagrange method. Lecture Notes in Physics*. Vol. 238. New York: Springer-Verlag.

DeVore, C.R., J.H. Gardner, J.P. Boris, and D. Mosher. 1984. Hydrodynamic simulations of light ion beam-matter interactions: Ablative acceleration of thin foils. *Laser Particle and Beams* 2:227–243.

Dukowicz, J.K., and J.W. Kodis. 1987. Accurate conservative remapping (rezoning) for arbitrary Lagrangian-Eulerian computations. *SIAM Journal of Scientific and Statistical Computing* 8(3):305–321.

Duque, E. 1994. A structured/unstructured embedded flow solver for helicopter rotor flows. In *American Helicopter Society 50th Annual Forum Proceedings*, Alexandria, VA., Vol II, 1249–1258.

Edwards, M. 1996. Elimination of adaptive grid interface errors in the discrete cell centered pressure equation. *Journal of Computational Physics* 126:356–372.

Eiseman, P.R. 1985. Grid generation for fluid mechanics computations. *Ann. Rev. Fluid Mech.* 17:487–522.

Emery, M.H., A.M. Landsberg, F. Felker, and C.T. Dyka. 1997. The dynamic virtual cell embedding technique (DVCE) for coupling hydrodynamic and structural mechanics codes. In *Proceedings of the Pressure Vessels and Piping Division Conference (Structures Under Extreme Loading Conditions)*. New York, New York American Society of Mechanical Engineers.

Finlayson, B.A., and L.E. Scriven. 1966. The method of weighted residuals – A review. *Appl. Mech. Rev.* 19:735–748.

Frisch, U., B. Hasslacher, and Y. Pomeau. 1986. A lattice gas automaton for the Navier-Stokes equation. *Physical Review Letters* 56:1505–1508.

Fritts, M.J., and J.P. Boris. 1979. The Lagrangian solution of transient problems in hydrodynamics using a triangular mesh. *Journal of Computational Physics* 31:173–215.

George, P.L. 1991. *Automatic mesh generation*. New York: Wiley.

Gottlieb, D., and S.A. Orszag. 1977. *Numerical analysis of spectral methods, theory and applications*. Philadelphia: SIAM.

Harlow, F.H., and J.F. Welch. 1965. Numerical calculation of time-dependent viscous incompressible flow of fluid with free surface. *Physics of Fluids* 8:2182–2189.

Hirt, C.W. 1970. An arbitrary Lagrangian-Eulerian computing technique. In *Proceedings of the Second International Conference on Numerical Methods in Fluid Dynamics*. Berkeley, Calif., Berlin: Springer-Verlag.

Kao, K.-H., M.-S. Liou, and C.-Y. Chow. 1994. Grid adaptation using Chimera composite overlapping meshes. *AIAA Journal* 32:942–949.

Khokhlov, A.M. 1998. Fully threaded tree algorithms for adaptive mesh refinement fluid dynamic simulations. *Journal of Computational Physics* 143:519–543.

Khokhlov, A.M., and E.S. Oran. 1999. Detonation initiation in a flame brush: The role of hot spots. *Combustion and Flame* 119:400–416.

Knupp, P., and S. Steinberg. 1993. *Fundamentals of grid generation*, Boca Raton, Fla.: CRC Press.

Kravchenko, A.G., P. Moin, and R. Moser. 1996. Zonal embedded grids for numerical simulations of wall-bounded turbulent flows. *Journal of Computational Physics* 127:412–423.

Lambrakos, S.G., and J.P. Boris. 1987. Geometric properties of the monotonic Lagrangian grid algorithm for near neighbor calculations. *Journal of Computational Physics* 73:183–202.

Landsberg, A.M., and J.P. Boris. 1997. The virtual cell embedding method: A simple approach for gridding complex geometries. AIAA paper 97-1982. Reston, VA: American Institute of Aeronautics and Astronautics.

Landsberg, A.M., J.P. Boris, W. Sandberg, and T.R. Young. 1993. Naval ship superstructure design: Complex three-dimensional flows using an efficient parallel method. In *High performance computing 1993: Grand challenges in computing*, ed. A.M. Tentner, 15–20. San Diego: SCS.

Landsberg, A.M., J.P. Boris, T.R. Young, and R.J. Scott. 1995. Computing complex shocked flows through the Euler equations. In *Shock waves @ Marseille* I, eds. R. Brun and L.Z. Dumitrescu, 421–426. Berlin: Springer-Verlag. (Presented at 19th International Symposium on Shock Waves, University of Provence, Marseille, 26–30 July 1993).

Lind, C.A., J.P. Boris, E.S. Oran, W.J. Mitchell, and J.L. Wilcox. 1997. The response of an open air detonation facility to blast loading. In *Proceedings of the 1997 Structures Under Extreme Loading Conditions Symposium*. PVP-vol. 351, ed. Y.S. Shin, 109–125. Orlando, Fla.: ASME.

———. 1998. The response of a partially confined detonation facility to blast loading. *Journal of Pressure Vessel Technology* 120:306–312.

Löhner, R. 1987. An adaptive finite element scheme for transient problems in CFD. *Computer Methods in Applied Mechanics and Engineering* 61:321–338.

———. 1989. Adaptive H-refinement on 3-D unstructured grids for transient problems. AIAA paper 89-0365. Reston, VA: American Institute of Aeronautics and Astronautics.

Löhner, R., and J. Baum. 1990. Numerical simulation of shock interation with complex geometry three-dimensional structures using a new adaptive H-refinement scheme on unstructured grids. AIAA paper 90-0700. Reston, VA: American Institute of Aeronautics and Astronautics.

Löhner, R., and P. Parikh. 1988. Three-dimensional grid generation by the advancing front method. *International Journal of Numerical Methods in Fluids* 8:1135–1149.

Maple, R.C., and D.M. Belk. 1994. Automated setup of blocked, patched, and embedded grids in the Beggar flow solver. In *Numerical grid generation in computational fluid simulation and related fields*, 305–314. Swansea, England: Pineridge Press.

Mavriplis, D.J. 1997. Unstructured grid techniques. *Ann. Rev. Fluid Mech.* 29:473–514.

Meakin, R.L., and N. Suhs. 1989. Unsteady aerodynamic simulation of multiple bodies in relative motion. AIAA paper 89-1996, presented at AIAA 9th Computational Fluid Dynamics Conference, Buffalo N.Y.

———. 1991. A new method for establishing intergrid communications among systems of overset grids. AIAA paper 91-1586–CP. Reston, VA: American Institute of Aeronautics and Astronautics.

———. 1995. An efficient means of adaptive refinement within systems of overset grids. AIAA paper 95-1722. Reston, VA: American Institute of Aeronautics and Astronautics.

Melton, J.E., M.J. Berger, M.J. Aftosmis, and M.D. Wong. 1995. 3D applications of a Cartesian grid Euler method. AIAA paper 95-0875. Reston, VA: American Institute of Aeronautics and Astronautics.

Mohr, G.A. 1992. *Finite Elements for Solids, Fluids, and Optimization*. Oxford: Oxford University Press.

Monaghan, J.J. 1985. Particle methods for hydrodynamics. *Comput. Phys. Rep.* 3:422–433.

———. 1988. Particle methods for hydrodynamics. *Comp. Phys. Comm.* 48:89–96.

Morgan, K., and J. Peraire. 1998. Unstructured grid finite-element methods for fluid mechanics. *Reports on Progress in Physics* 61:569–638.

Nicolaides, R.A. 1993. The covolume approach to computing incompressible flows. In *Incompressible computational fluid dynamics*, 295–334. New York: Cambridge University Press.

Nikiforakis, N., S.J. Billet, E. Boden, and E.F. Toro. 1999. Application of adaptive mesh refinement to global atmospheric modelling, submitted to *Quarterly Journal of the Royal Meteorological Society*.

Nikiforakis, N., and E.F. Toro. 1999. Riemann-problem-based methods for global atmospheric off-line simulations, submitted to *Quarterly Journal of the Royal Meteorological Society*.

Oran, E.S., T.R. Young, and J.P. Boris. 1978. Application of time-dependent numerical methods to the description of reactive shocks. In *Seventeenth Symposium (International) on Combustion*, 43–53. Pittsburgh, Pa.: The Combustion Institute.

Oran, E.S., T.R. Young, J.P. Boris, and A. Cohen. 1982a. Weak and strong ignition, I. Numerical simulations of shock tube experiments. *Combustion and Flame* 48:135–148.

Oran, E.S., T.R. Young, J.P. Boris, J.M. Picone, and D.H. Edwards. 1982b. A study of detonation structure: The formation of unreacted pockets. In *Nineteenth Symposium (International) on Combustion*, 573–582. Pittsburgh, Pa.: The Combustion Institute.

Patera, A.T. 1984. A spectral element method for fluid dynamics, laminar flow in a channel expansion. *Journal of Computational Physics* 54:468–488.

Peskin, C.S. 1977. Numerical analysis of blood flow in the heart. *Journal of Computational Physics* 25:220–252.

Quirk, J.J. 1994. An alternative to unstructured grids for computing gas dynamics flows around arbitrarily complex two-dimensional bodies. *Computers and Fluids* 23(1):125–142.

Ramamurti, R., K. Kailasanath, and R. Löhner. 1992. Numerical simulation of unsteady flow in a nozzle. In *Proceedings of the 1992 JANNAF Propulsion Meeting*, CPIA Publication 580. Laurel, Md.: CPIA.

Ramamurti, R., R. Löhner, and W.C. Sandberg, 1999. Computation of 3-D unsteady flow past deforming geometries. *International Journal of Computational Fluid Dynamics* 13:83–99

Roache, P.J. 1997. Quantification of uncertainty in computational fluid dynamics. *Ann. Rev. Fluid Mech.* 29:123–160.

Schneiders, R., and R. Bunten. 1995. Automatic grid generation of hexahedral finite element meshes. *Comput. Aided Geom. Descrip.* 12:673–707.

Simon, B. 1993. *The statistical mechanics of lattice gases. Vol. I*. Princeton, N.J.: Princeton University Press.

Sinkovits, R.S., J.P. Boris, and E.S. Oran. 1994. A technique for regularizing the structure of a monotonic Lagrangian grid. *Journal of Computational Physics* 108:368–372.

Swegle, J.W., S.W. Attaway, M.W. Heinstein, F.J. Mello, and D.L. Hicks. 1994. *An analysis of smoothed particle hydrodynamics*. Albuquerque, N.Mex.: Sandia National Laboratories, Report SAND93-2513.

Tannehill, J.C., D.A. Anderson, and R.H. Pletcher. 1997. *Computational fluid mechanics and heat transfer*. Washington, D.C.: Taylor and Francis.

Thompson, J.F., and B. Hamann. 1997. A survey of grid generation techniques and systems with emphasis on recent developments. *Surv. Math. Ind.* 6:289–310.

Thompson, J.F., Z.U.A. Warsi, and C.W. Mastin. 1985. *Numerical grid generation – Foundations and applications*. New York: Elsevier Science Publishing Co.

Tramel, R.W., and R.H. Nichols. 1997. A highly efficient numerical method of overset-mesh moving-body problems. AIAA paper 97-2040, presented at 13th Computational Fluid Dynamics Conference, Snowmass, Colo.

Wang, Z.J. 1995. A conservative interface algorithm for overlapping grids. *Journal of Computational Physics* 122:96–106.

Warsi, Z.U.A., and J.F. Thompson. 1990. Applications of variational methods in fixed and adaptive grid generation. *Computers and Mathematics with Applications* 19:31–41.

Yerry, M.A., and M.S. Shephard. 1985. Automatic mesh generation for three-dimensional solids. *Comput. Struct.* 20:31–39.

Young, D.P., R.G. Melvin, M.B. Bieterman, F.T. Johnson, S.S. Sament, and J.E. Bussoletti. 1991. A locally refined rectangular finite element method: Application to computational fluid dynamics and computational physics. *Journal of Computational Physics* 92:1–66.

Young, T.R., A.M. Landsberg, and J.P. Boris. 1993. Implementation of the full 3D FAST3D (FCT) code including complex geometry on the Intel iPSC/860 parallel computer. In *High performance computing 1993: Grand challenges in computing*, ed. A.M. Tentner, 143–148. San Diego: SCS.

Zienkiewicz, O.C., and K. Morgan. 1983. *Finite elements and approximation*. New York: John Wiley and Sons.

7

Diffusive Transport Processes

The term *diffusive transport* encompasses molecular diffusion, viscosity, thermal conduction, thermal diffusion, and radiation transport. These parts of equations (2–1.1) through (2–1.18) are represented by the expressions

$$\frac{\partial n_i}{\partial t} = -\nabla \cdot n_i \mathbf{v}_{di}, \tag{7–0.1}$$

$$\frac{\partial \rho \mathbf{v}}{\partial t} = -\nabla \cdot \delta \mathbf{P}, \tag{7–0.2}$$

and

$$\frac{\partial E}{\partial t} = -\nabla \cdot (\delta \mathbf{q} + \mathbf{q}_r). \tag{7–0.3}$$

Here $\delta\mathbf{P}$ and $\delta\mathbf{q}$ symbolically indicate those parts of $\mathbf{P}$ and $\mathbf{q}$ in the full set of reactive-flow conservation equations involving momentum, heat, and molecular transport. Each equation written above is a divergence of a flux. Each of these fluxes, though not explicitly displayed, is related to a local gradient. The convection terms are omitted from these equations to allow us to focus on the diffusive transport processes.

In equation (7–0.1), $\mathbf{v}_{di}$ is the diffusion velocity of species i, defined as

$$\mathbf{v}_{di} = \mathbf{v}_i - \mathbf{v} \tag{7–0.4}$$

where $\mathbf{v}$ is the fluid velocity averaged over all species and $\{\mathbf{v}_i\}$ are the velocities of the different fluid species in the same frame of reference as $\mathbf{v}$.

The diffusion velocities $\{\mathbf{v}_{di}\}$ are found by inverting the matrix equation

$$\mathbf{G}_i = \sum_k \frac{n_i n_k}{N^2 D_{ik}} (\mathbf{v}_{dk} - \mathbf{v}_{di}), \tag{7–0.5}$$

where the source terms $\{\mathbf{G}_i\}$ are defined as

$$\mathbf{G}_i \equiv \nabla\left(\frac{n_i}{N}\right) - \left(\frac{\rho_i}{\rho} - \frac{n_i}{N}\right)\frac{\nabla P}{P} - K_i^T \frac{\nabla T}{T}. \tag{7–0.6}$$

Equation (7–0.1) describes diffusion, although it superficially resembles convection because the fluxes are linear in the spatial gradients through equations (7–0.5) and (7–0.6). The diffusion velocities are defined relative to the net momentum of the multispecies mixture, and thus are constrained to carry no net momentum,

$$\sum_{i=1}^{N_s} \rho_i \mathbf{v}_{di} = 0. \tag{7–0.7}$$

This constraint is the additional relation needed to remove the singularity arising in the inversion of equation (7–0.5), whose determinant is zero.

In equation (7–0.2), the divergence is taken of the pressure tensor, $\mathbf{P}$, minus the local diagonal contribution, $P(N, T)\mathbf{I}$,

$$\delta\mathbf{P} = \left(\frac{2}{3}\mu_m - \kappa\right)(\nabla \cdot \mathbf{v})\mathbf{I} - \mu_m[(\nabla\mathbf{v}) + (\nabla\mathbf{v})^T]. \tag{7–0.8}$$

These are the diffusive viscous terms, in which μ_m and κ multiply gradients of the velocity.

Equation (7–0.3) represents diffusion of heat due to gradients in temperature and pressure. Subtracting the zeroth-order convective energy flux gives

$$\delta\mathbf{q}(N, T) = -\lambda_m \nabla T + \sum_i n_i h_i \mathbf{v}_{di} + P \sum_i K_i^T \mathbf{v}_{di}. \tag{7–0.9}$$

The quantity $\mathbf{q}_r$ is the heat flux due to radiation.

This chapter describes methods for solving these equations and some of the difficulties associated with these methods. It also presents generic formulas for the diffusion coefficients appropriate for combustion. Radiation transport, which is sometimes treated as a diffusive process, is treated separately in Chapter 13.

7–1. The Diffusion Equation

7–1.1. The Physical Origin of Diffusive Effects

Diffusion is a macroscopic effect arising from the microscopic thermal motions of particles. The particles bounce back and forth between collisions in a random walk about the average fluid velocity, $\mathbf{v}$. Electromagnetic radiation, in the form of photons, can also be repeatedly absorbed and re-emitted as it interacts with the molecules. Thus radiation also random walks as it goes through matter. These random motions about an average act to spread gradients, mix materials at sharp interfaces, create motion in the fluid, and transport mass, momentum, and energy. Thus diffusion processes are important macroscopically, despite the fact that no net mass flow is involved.

The generic representation of diffusion considered here is the second-order partial differential equation

$$C(\rho, \mathbf{x}, t)\frac{\partial \rho}{\partial t} = \nabla \cdot D(\rho, \mathbf{x}, t)\nabla\rho + S(\mathbf{x}, t) \tag{7–1.1}$$

where ρ indicates a generic conserved variable. The quantities C and D may be constant or may depend on the system variables. When the energy equation is rewritten as

a temperature equation, ρ is a temperature and the function C is a specific heat. When the ρ represents the momentum, C is the density weighting that relates the velocity to the momentum. For species diffusion equations, ρ represents the density of the individual species, a vector of length N_s at each spatial location. In this case the diffusion coefficient is a tensor, $\mathbf{D}$, and equation (7–1.1) becomes

$$\frac{\partial \boldsymbol{\rho}}{\partial t} = \nabla \cdot \mathbf{D}(\boldsymbol{\rho}, \mathbf{x}, t)\nabla \boldsymbol{\rho}. \tag{7–1.2}$$

The diffusion equation is a simple macroscopic model of the complicated dynamics of many microscopic particle interactions. This model works so well that many different microscopic phenomena are represented by the same macroscopic form for the equation. These phenomena include particle diffusion in gases and liquids; momentum diffusion in gases and liquids; thermal energy diffusion in gases, liquids, and solids; radiation transport in optically thick media; turbulent transport on large spatial scales; the repeated averaging of observations; and numerical smoothing from cell uncertainty. In each case, the diffusion equation arises from an average of many interacting degrees of freedom. The averaging process, that results in an equation of the form of equation (7–1.2), collapses the more complicated physics occurring on small space and fast time scales into a simpler macroscopic model. Such a macroscopic representation is imperfect in that it has lost information. The separation between the microscopic and macroscopic scales can never be complete, and higher-order effects from the microscopic physics may extend into the macroscopically averaged fluid equations. Such effects, although neglected in the diffusion approximation, may be substantial.

Thus it can be no surprise that the mathematical formulas that represent diffusion as a continuum process break down in some regimes. This can be insidious because even though the mathematical solution based on equation (7–1.2) may appear to evolve in a reasonable way, the answer may not reproduce the physically correct behavior of the system.

For example, the mathematical expression for molecular diffusion is not physically valid for scale lengths comparable to the mean free path between particle collisions. These errors become small, however, when gradients extend over many mean free paths. It is more surprising that the diffusion equation also breaks down at large distances and finite times. To help understand the intrinsic errors in the diffusion equation model, we now examine a δ-function test problem whose analytic solution encompasses all wavelengths and time scales, and therefore can be evaluated in various limits.

7–1.2. The δ-Function Test Problem

A finite amount of a conserved quantity, with density $\rho(\mathbf{x}, t)$, is deposited at $\mathbf{x} = \mathbf{x}_o$ at time $t = t_o$. It spreads diffusively in time, and the value of ρ at $\mathbf{x} = \mathbf{x}_o$ decreases. The total integral of ρ over all space cannot change with time. We assume that there are no sources and that D is constant in time and space. The solution for $\rho(\mathbf{x}, t)$ at later times becomes Gaussian, and the characteristic length scale $k^{-1}(t)$ and amplitude $\rho(t)$ vary with time.

Substituting a solution of the form

$$\rho(x, t) = \rho(t)e^{-k^2(t)x^2} \tag{7–1.3}$$

into equation (7–1.2) gives two equations that must be satisfied,

$$k(t)\frac{d}{dt}k(t) = -2k^4(t)D, \tag{7–1.4}$$

and

$$\frac{d}{dt}\rho(t) = -2k^2(t)D\rho(t). \tag{7–1.5}$$

The solution of these two equations gives a closed-form result,

$$\rho(x,t) = \rho(t_1)\left(\frac{t}{t_1}\right)^{-\frac{1}{2}} \exp\left(-\frac{x^2}{4D(t-t_o)}\right), \tag{7–1.6}$$

where the characteristic wavenumber, $k(t)$, evolves according to

$$k^2(t) = \frac{1}{4D(t-t_o)}. \tag{7–1.7}$$

The time $t_1 > t_o$ is the starting time when the density at $x = 0$ is $\rho(t_1)$.

The analytic solution for ρ in d dimensions ($d = 1$, 2, or 3) is

$$\rho(\mathbf{x},t) = \rho_o\left(\frac{k}{\sqrt{\pi}}\right)^d e^{-k^2(t)\ (\mathbf{x}-\mathbf{x}_o)^2}. \tag{7–1.8}$$

We call this a δ-function solution because the limit of equation (7–1.8), as t approaches t_o, is one definition of the singular Kronecker δ-function,

$$\delta(\mathbf{x}-\mathbf{x}_o) = \lim_{k\to\infty}\left(\frac{k}{\sqrt{\pi}}\right)^d e^{-k^2(\mathbf{x}-\mathbf{x}_o)^2}. \tag{7–1.9}$$

For all times $t > t_o$, this solution is Gaussian in shape and is also conservative because the area under the curve $\rho(\mathbf{x}, t)$ is constant in time. A decaying Gaussian is shown in Figure 7.1.

When $t = t_o$, the solution is singular and $k^2(t)$ from equation (7–1.7) is infinite. This corresponds to all of ρ, in this case

$$\rho_o = \int_{-\infty}^{\infty} d\mathbf{x}\ \rho(\mathbf{x}, t_o), \tag{7–1.10}$$

deposited initially at $\mathbf{x}_o$. The wavenumber k is well defined for all times greater than t_o. The distribution of ρ is still very localized, essentially a δ-function when t is slightly larger than t_o, but the profiles are finite.

Even for more complicated and dispersed initial conditions, the solutions approach a Gaussian asymptotically. When a structured initial distribution of $\rho(\mathbf{x}, t_o)$ is localized in a region of space, it can be represented as a superposition of δ-functions. Because the diffusion equation is linear, each δ-function component individually follows equation (7–1.3) or (7–1.8) at subsequent times. The composite solution also approaches a Gaussian when $1/k(t)$ becomes larger than L_o, characterizing the size of the region of initial sources.

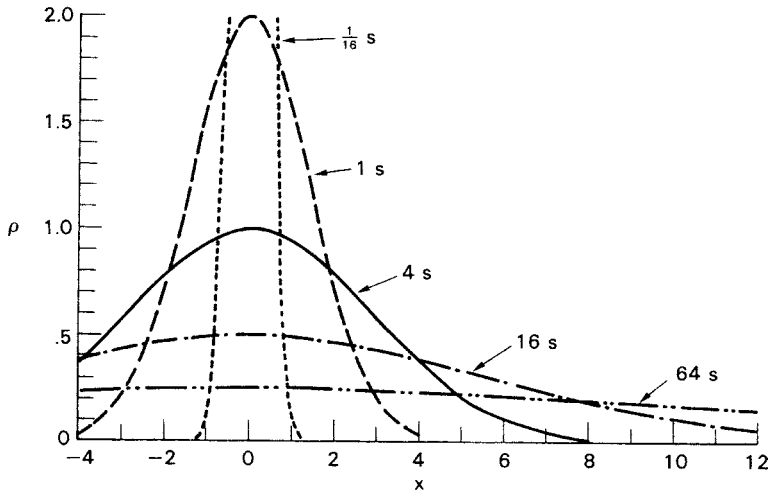

Figure 7.1. A decaying Gaussian solution with $D = 1$ cm/s^2. The integrated area under each curve has the same value.

The composite solution is

$$\rho(\mathbf{x}, t) = \int_{-L_o}^{L_o} d\mathbf{x}_o \, \rho(\mathbf{x}_o, 0) \left(\frac{k(t)}{\sqrt{\pi}} \right)^d e^{-k^2(t) \, (\mathbf{x}-\mathbf{x}_o)^2}. \tag{7–1.11}$$

The integral limits signify that the initial source distribution is localized within a distance L_o of the origin and involves only a finite amount of ρ, given by

$$\rho_o = \int_{-L_o}^{L} d\mathbf{x}_o \, \rho(\mathbf{x}_o, 0). \tag{7–1.12}$$

After enough time elapses, $k(t)L_o$ becomes less than unity and each δ-function element of the overall solution spreads until the initial source region is overlapped by all the δ-function components. The equation for $\rho(\mathbf{x}, t)$, equation (7–1.11), approaches the Gaussian solution for ρ_o initially placed at the center of mass of the source distribution $\rho(\mathbf{x}_o, 0)$. Because $k(t)$ decreases as time increases, for any distance $|\mathbf{x}|$ there is always a finite time t_x after which $k|\mathbf{x}| < \frac{1}{2}$,

$$t_x = t_o + \frac{L_o|\mathbf{x}|}{2D}. \tag{7–1.13}$$

For times less than t_x, diffusion has not spread the distributions sufficiently to eliminate the influence of the initial conditions. After t_x has elapsed, however, there is an expanding shell about the source region. The profile inside this shell, $\rho(\mathbf{x}, t)$, is essentially indistinguishable from a composite one-Gaussian source in which the equivalent amount of ρ_o is deposited at the center. Outside this shell the solution is exponentially small, but large differences are possible between the lumped-source solution and the multi-Gaussian solution with a nonlocalized initial density profile.

The important point is that initially localized diffusion problems approach profiles where deviations from the appropriate Gaussian decrease rapidly at any location. These

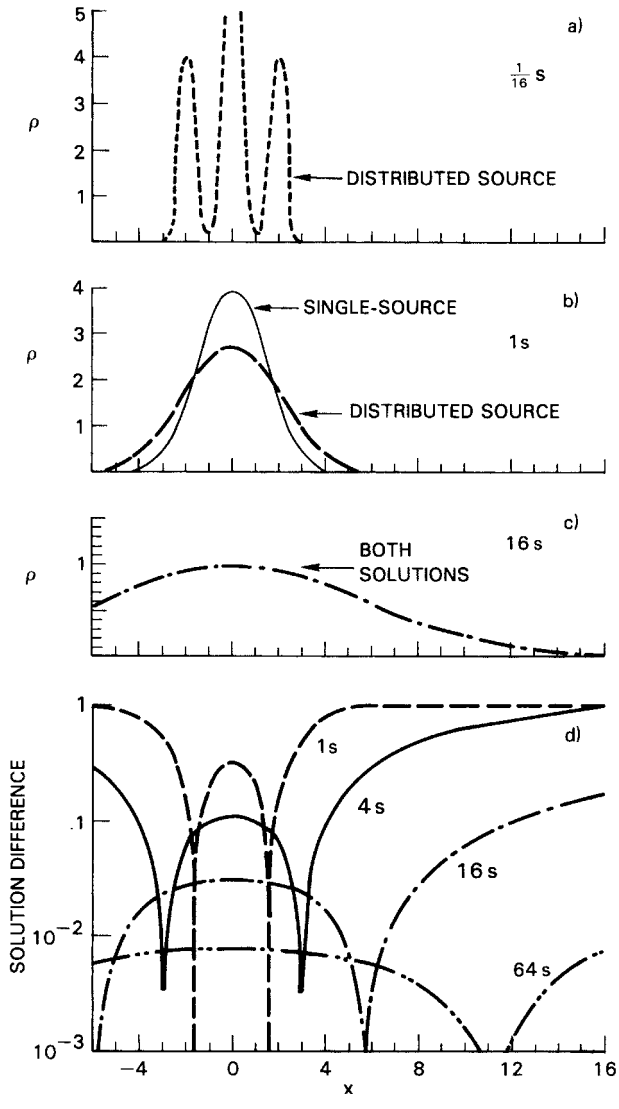

Figure 7.2. (a) Three δ-function source after 1/16 s. (b) Three δ-function source and equivalent composite δ-function source after 1 s. (c) The two solutions after 16 s are identical on the scale of the figure. (d) The difference between the two solutions at several times.

deviations are relatively large only in the exponentially small tail of the solution, where the diffusion equation breaks down anyway. Figure 7.2 shows the evolution of such a superposition graphically. Three δ-functions are initially located at $x = -1$, 0, and 1 cm with twice the amount of ρ in the central Gaussian as in the other two. The exact solution and the single composite Gaussian solution are shown at later times to show the merging of the separate Gaussians. The lower panel of the figure shows the difference between the lumped-parameter one-Gaussian solution and the three-Gaussian solution.

The tendency of a localized diffusing system to approach a Gaussian has important numerical implications. If errors in the solution tend to decrease in time as the calculation proceeds, the solution becomes more accurate rather than less accurate. The

short-wavelength components of the solution diffuse the fastest, and their amplitudes quickly approach zero if there are no sources of short-wavelength structures. This property is useful because numerical simulations using finite spatial grids usually have the worst errors at short wavelengths. If physical diffusion is large enough, it actually reduces the continuing short-wavelength errors generated by numerical representations with finite resolution.

7–1.3. Short- and Long-Wavelength Breakdowns

In Chapter 4, the diffusion flux was defined as

$$\mathbf{f}_D \equiv -D\,\nabla\rho, \tag{7–1.14}$$

proportional to the gradient of the dependent fluid variable $\rho(\mathbf{x}, t)$. On the scale of particle mean free paths, this gradient is relatively meaningless and thus the flux, given by equation (7–1.14), gives a solution that is too smooth. The diffusion equation is derived by taking macroscopic averages over many individual particles; it is not surprising that the representation fails at short scale lengths. Thus it is often attractive to look for higher-order continuum corrections to the diffusion equation at these short scales. Unless these corrections contain a high-frequency source component, however, they cannot represent the qualitative behavior of the random-walk process at short scales.

Another regime in which the diffusion equation fails is far from fluctuating sources at relatively long times after the sources have been deposited. Consider ρ from a single source diffusing through space. Any amount of ρ appearing at a distant location $\mathbf{x}$ arrives there by traveling for a finite time through all intermediate locations. An average diffusive transport velocity can be calculated from the δ-function Gaussian solution,

$$\mathbf{v}_D = \frac{\mathbf{f}_D}{\rho} = \frac{(\mathbf{x} - \mathbf{x}_o)}{2(t - t_o)}, \tag{7–1.15}$$

where $\mathbf{v}_D$ is the diffusion velocity.

Thus at long wavelengths it appears that the propagation velocity does not depend on the diffusion constant! This is the macroscopic consequence of the fact that the thermal velocity, which relates to the diffusive propagation speed, does not depend on the mean free path. Hence the diffusion velocity is constant. This result is curious because equation (7–1.15) indicates that for any finite time, the velocity of diffusive transport becomes arbitrarily large for $|\mathbf{x} - \mathbf{x}_o|$ approaching ∞. The exponentially small tails that a linear diffusion equation generates at large distances actually move far too fast for the random walk of the particles. This is indicated by the far-field limit of Figure 7.2d.

It is tempting to try to identify this superfast diffusive motion with the very fast particles in the tails of the thermal distribution, but this explanation is incorrect. Some of the superfast particles would have to exceed the speed of light when $\mathbf{x}$ is far enough from $\mathbf{x}_o$. For example, the distance between air molecules at standard temperature and pressure is about 30 Å and their mean free path is about 1,000 Å. There are about 5×10^9 collisions per particle per second. In one collision time, 2×10^{-10} s, the continuum Gaussian solution has particles moving at the speed of light at a distance of 12 cm from the source location. Fortunately, they are slowing down; they had to travel at an average speed of 6×10^{10} cm/s

to get there! This nonphysical behavior of the analytic solution results from the inability of the Gaussian to represent the correct long-wavelength behavior of the particles.

7–1.4. Numerical Diffusion and Instability

Numerical diffusion refers to errors that arise both from truncation in the algorithms used and the discrete nature of the representation. In a computation, numerical diffusion behaves like physical diffusion at scale lengths of a few cells and longer, but its origin is not physical. Thus results which are quantitatively incorrect can appear reasonable. The term *numerical diffusion* contributes additional confusion. By referring to the unwanted phenomenon of excess numerical damping as "numerical diffusion," we invest it with a modicum of legitimacy that it does not deserve.

There is an unavoidable uncertainty about where in a computational cell the fluid actually is, and this is itself a major source of numerical diffusion. The average density in a cell of finite volume does not contain information about how the density is distributed within the cell. This uncertainty is particularly important in solving the continuity equation and is discussed further in Chapter 8.

Numerical instability often appears in a computed solution as if a diffusion equation were being run backward, or as if one of the diffusion coefficients were inadvertently made negative. Too large a timestep or too large a gradient in a spatial derivative can change the sign of the damping term in the numerical amplification factor for the algorithm. These errors grow quickly from the initial conditions. A smoothly varying function begins to show small fluctuations, which rapidly become spikes. Because the short wavelengths grow fastest, the location of the biggest spike is usually a good indicator of where the stability condition is first violated. Because of finite resolution and nonlinear interactions in the overall model, waves of spikes propagate out from the initial site at subsequent times.

7–2. Integration of the Diffusion Equation

7–2.1. A Finite-Difference Formula in One Dimension

Section 4–3 introduced a finite-difference approximation to the diffusion operator and solved the problem of the diffusive smoothing of a periodic function. Explicit, semi-implicit, and fully implicit algorithms were discussed, and the results were compared to the corresponding analytic solution. We assumed constant timesteps, a uniformly spaced one-dimensional grid, and a constant diffusion coefficient. This section considers one-dimensional diffusion and treats spatial variations in the cell sizes and diffusion coefficients.

Consider the diffusion of $\rho(x, t)$ in a rigid duct whose cross-sectional area, $A(x)$, varies with x, as shown schematically in Figure 7.3. A conservative finite-difference form of the diffusion equation with a variably spaced grid and variable diffusion coefficients is given by

$$\Lambda_i \frac{\left(\rho_i^n - \rho_i^{n-1}\right)}{\Delta t} = A_{i+\frac{1}{2}} D_{i+\frac{1}{2}} \frac{(\overline{\rho}_{i+1} - \overline{\rho}_i)}{(x_{i+1} - x_i)} - A_{i-\frac{1}{2}} D_{i-\frac{1}{2}} \frac{(\overline{\rho}_i - \overline{\rho}_{i-1})}{(x_i - x_{i-1})} + \Lambda_i S_i, \qquad i = 1, 2, \ldots, N. \quad (7\text{–}2.1)$$

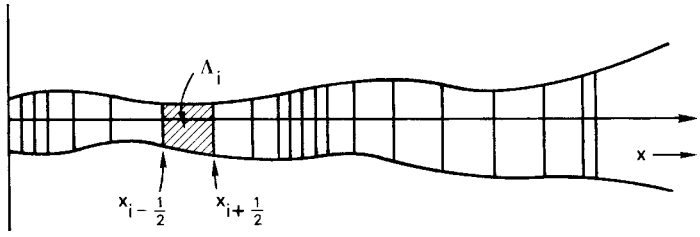

Figure 7.3. Schematic of a rigid duct with variable cross-sectional area.

Here $\{\Lambda_i\}$ are the volumes of cell i, $\{A_{i+\frac{1}{2}}\}$ are the areas of the cell interfaces, and $\{D_{i+\frac{1}{2}}\}$ are the diffusion coefficients evaluated at the cell interfaces. The $\{S_i\}$ are source or sink terms representing the effects of chemical reactions. The bars over the values of ρ on the right side indicate that an average in time can be taken, giving centered, semi-implicit, or fully implicit schemes to improve the order of accuracy of the time integration. In equation (7–2.1), cell-interface locations are denoted $\{x_{i+\frac{1}{2}}\}$, with $i = 0, 1, \ldots, N$. The left boundary of the computational region is $x_{\frac{1}{2}}$ and the right boundary is $x_{N+\frac{1}{2}}$.

This approximation to the diffusion equation conserves the total amount of ρ. Diffusive flux is defined at the cell interfaces, so the exact amount of ρ leaving one cell is automatically added to the next cell. Conservation can be shown by summing equation (7–2.1) over all cells to obtain

$$\sum_{i=1}^{N} \Lambda_i \rho_i^n = \sum_{i=1}^{N} \Lambda_i \rho_i^{n-1} + \sum_{i=1}^{N} S_i + \Delta t\left[A_{N+\frac{1}{2}} D_{N+\frac{1}{2}} (\nabla \overline{\rho})_R - A_{\frac{1}{2}} D_{\frac{1}{2}} (\nabla \overline{\rho})_L\right]. \tag{7–2.2}$$

The amount of ρ in the computational domain at the new time is equal to the amount at the old time plus any that comes in from the boundaries minus any that goes out plus the sum of all the source terms.

7–2.2. Implicit Solutions Using Tridiagonal Matrix Inversions

When $\overline{\rho}$ in equation (7–2.1) is replaced by ρ^{n-1}, the finite-difference formula is explicit, and thus solving for ρ^n is straightforward. Values of ρ at the new time are a function of three values at the previous time. The effects of a local change propagate at one cell per timestep.

When the expression for $\overline{\rho}$ includes some of the ρ^n in addition to ρ^{n-1}, the algorithm is at least partially implicit. For example, if

$$\overline{\rho}_i = \theta \rho_i^n + (1 - \theta)\rho_i^{n-1} \tag{7–2.3}$$

is substituted into equation (7–2.1), the equation can be rearranged into matrix form,

$$\mathbf{M} \cdot \boldsymbol{\rho}^n = \mathbf{s}^o. \tag{7–2.4}$$

Here $\mathbf{M}$ is a tridiagonal matrix of coefficients and $\mathbf{s}^o$ is a vector containing the

inhomogeneous and source terms,

$$
\begin{aligned}
M_{i,i-1} &= -\theta B_{i-1} \\
M_{i,i} &= \frac{\Lambda_i}{\Delta t} + \theta(B_{i+1} + B_{i-1}) \\
M_{i,i+1} &= -\theta B_{i+1} \\
s_i^o &= (1-\theta)B_{i-1}\rho_{i-1}^{n-1} + \left[\frac{\Lambda_i}{\Delta t} - (1-\theta)(B_{i+1} + B_{i-1})\right]\rho_i^{n-1} \\
&\quad + (1-\theta)B_{i+1}\rho_{i+1}^{n-1} + S_i,
\end{aligned}
\tag{7–2.5}
$$

where

$$
\begin{aligned}
B_{i-1} &= \frac{A_{i-\frac{1}{2}} D_{i-\frac{1}{2}}}{x_i - x_{i-1}} \\
B_{i+1} &= \frac{A_{i+\frac{1}{2}} D_{i+\frac{1}{2}}}{x_{i+1} - x_i}.
\end{aligned}
\tag{7–2.6}
$$

The $\boldsymbol{\rho}^n$ is a vector of all of the values of $\{\rho_i\}$ at the new time. Note that the coefficients $C(\rho, \mathbf{x}, t)$ in equation (7–1.1) may be included in the Λ_i if they are constant during a timestep.

7–2.3. Large Variations in Δx and D

The advantages and problems associated with using variably spaced zones are discussed in Chapter 6. The motivation for considering such methods is that they improve resolution where there are significant changes in the flow field and they reduce computational effort and cost when only slow variations are present. There are, however, numerical problems with variably spaced grids. For example, using a variably spaced grid, as equation (7–2.1) allows, does not automatically guarantee accuracy. When adjacent cell sizes vary significantly, the order of accuracy of an algorithm is often reduced and the truncation error increases.

There are practical ways to use variable spacing or rapidly varying diffusion coefficients and yet maintain enough accuracy. One approach is to keep the rate of change of the cell sizes small, so the truncation errors are small. The acceptable amount of variation from cell to cell varies with the particular physical problem and with the numerical algorithm. Sometimes it is possible to space the cells to ensure that spatial derivatives are of arbitrary order. This is often not practical, especially when high resolution is required to track regions in the flow with large gradients. The practical rule for variable spacing is to allow maximum changes of 10 to 15 percent from cell to cell.

Figure 7.4 shows several adjacent cells in which the density, indicated as point values at cell centers, and the diffusion coefficients, indicated as piecewise constant within each cell, vary significantly from cell to cell. Adjacent cell sizes can also be very different. Such situations usually occur at internal interfaces. An example is the interface between a relatively cool liquid droplet and warm background air where the thermal conductivity varies discontinuously across the interface. Even in single-phase media such situations

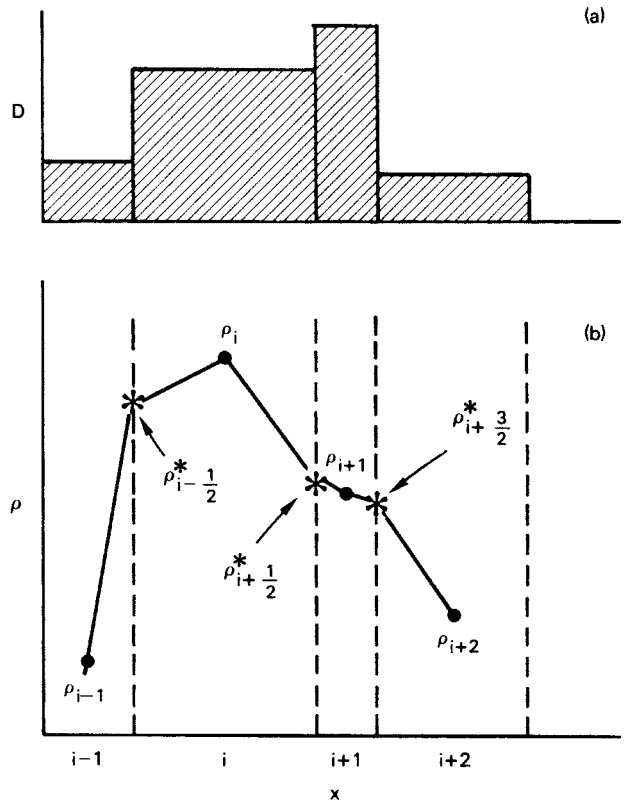

Figure 7.4. (a) Diffusion coefficient variations from cell to cell. (b) Values of the variable ρ at the cell centers (indicated by $\ldots, i-1, i, i+1, \ldots$) and cell interfaces (indicated by $\ldots, i-\frac{1}{2}, i+\frac{1}{2}, \ldots$).

occur in the vicinity of steep gradients where cell sizes and nonlinear diffusion coefficients vary rapidly.

As shown in the figure, the numerical approximation to such systems has very different states on opposite sides of an interface. Choosing average diffusion coefficients at the interface, $\{D_{i+\frac{1}{2}}\}$, should take these discontinuities into account. The trick to choosing this average is to define a fictitious value of ρ at the interface, called $\rho^*_{i+\frac{1}{2}}$ in the figure. This value guarantees that the flux F_-, which reaches the interface from the left, exactly matches the flux F_+, which leaves the interface on the right. This *flux-matching* condition ensures that the averaged value of ρ at the interface is physically meaningful. The condition that $F_+ = F_-$ determines $\rho^*_{i+\frac{1}{2}}$,

$$F_+ \equiv A_{i+\frac{1}{2}}\left(\frac{\overline{\rho}_{i+1} - \rho^*_{i+\frac{1}{2}}}{x_{i+1} - x_{i+\frac{1}{2}}}\right) D_{i+1} = F_- \equiv A_{i+\frac{1}{2}}\left(\frac{\rho^*_{i+\frac{1}{2}} - \overline{\rho}_i}{x_{i+\frac{1}{2}} - x_i}\right) D_i. \tag{7–2.7}$$

The factors of $A_{i+\frac{1}{2}}$ cancel and equation (7–2.7) can be solved for $\rho^*_{i+\frac{1}{2}}$ giving

$$\rho^*_{i+\frac{1}{2}} = \frac{\overline{\rho}_{i+1} D_{i+1}\left(x_{i+\frac{1}{2}} - x_i\right) + \overline{\rho}_i D_i\left(x_{i+1} - x_{i+\frac{1}{2}}\right)}{D_{i+1}\left(x_{i+\frac{1}{2}} - x_i\right) + D_i\left(x_{i+1} - x_{i+\frac{1}{2}}\right)}. \tag{7–2.8}$$

Note that the cell sizes and diffusion coefficients can vary extensively and $\rho^*_{i+\frac{1}{2}}$ still is between $\overline{\rho}_{i+1}$ and $\overline{\rho}_i$, a reasonable expectation of a weighted average.

Using this provisional value $\rho^*_{i+\frac{1}{2}}$, an average diffusion coefficient at the interface can be defined that incorporates the information in $\{\rho^*_{i+\frac{1}{2}}\}$. Inserting equation (7–2.8) into equation (7–2.7) and setting this equal to the flux $F_{i+\frac{1}{2}}$ gives

$$\begin{aligned} F_{i+\frac{1}{2}} &= A_{i+\frac{1}{2}} D_{i+\frac{1}{2}} \left(\frac{\overline{\rho}_{i+1} - \overline{\rho}_i}{x_{i+1} - x_i} \right) = F_+ \\ &= \frac{A_{i+\frac{1}{2}} D_i D_{i+1} (\overline{\rho}_{i+1} - \overline{\rho}_i)}{D_{i+1}\left(x_{i+\frac{1}{2}} - x_i\right) + D_i\left(x_{i+1} - x_{i+\frac{1}{2}}\right)}. \end{aligned} \tag{7–2.9}$$

The definition of the interface-averaged diffusion coefficients, $\{D_{i+\frac{1}{2}}\}$, given by equation (7–2.9), is

$$D_{i+\frac{1}{2}} = \frac{D_i D_{i+1} (x_{i+1} - x_i)}{D_{i+1}\left(x_{i+\frac{1}{2}} - x_i\right) + D_i\left(x_i - x_{i+\frac{1}{2}}\right)}. \tag{7–2.10}$$

Using these values of $\{D_{i+\frac{1}{2}}\}$ ensures the fewest inconsistencies in the fluxes at all of the interfaces when solving equation (7–2.1). Note that this formulation avoids using the provisional values $\{\rho^*_{i+\frac{1}{2}}\}$, so that tridiagonal matrix solvers can still be used to solve implicit forms of equation (7–2.1). This flux-matching trick is useful in many ways, including nonlinear diffusion and convection. This idea is used a number of times in this book.

7–3. Nonlinear Effects and Strong Diffusion

7–3.1. Nonlinear Diffusion Effects

Nonlinearities are impediments to solving the diffusion equation quickly and accurately. Such a nonlinearity exists if, for example, the diffusion coefficient $D(\rho, \mathbf{x}, t)$ in equation (7–1.1) depends on the current values of ρ. An important example of a nonlinear diffusion equation is derived from equation (7–0.3), considered as an equation for temperature, T. The thermal conduction coefficient $\lambda(T)$ usually increases with temperature at least as fast as $T^{\frac{1}{2}}$. Therefore, nonlinear thermal conduction often appears in the form

$$\frac{\partial T}{\partial t} = \frac{\partial}{\partial x}\lambda(T)\frac{\partial T}{\partial x} = \frac{\partial}{\partial x}\lambda_{\frac{1}{2}} T^{\frac{1}{2}} \frac{\partial T}{\partial x} = \frac{2\lambda_{\frac{1}{2}}}{3} \frac{\partial}{\partial x}\left(\frac{\partial T^{\frac{3}{2}}}{\partial x}\right). \tag{7–3.1}$$

Here it has been possible to bring the explicit temperature dependence of the diffusion coefficient inside the gradient operator. (In cgs units, the coefficient $\lambda_{\frac{1}{2}}$ has units of $\mathrm{K}^{-\frac{1}{2}}\ \mathrm{cm}^2\ \mathrm{s}^{-1}$.)

The heat flux is stronger than a linear function of the temperature. A thermal-conduction coefficient with the temperature raised to the $\frac{1}{2}$ or $\frac{3}{4}$ power is typical in gas-phase combustion and is important in flames. This is a weak nonlinearity when the power-law exponent is less than unity, but still the conduction becomes more important as the temperature increases. The electron thermal conductivity in a highly ionized plasma scales as $T^{\frac{5}{2}}$, and radiation transport scales as about T^4. In general, if $\lambda(T)$ is a monotonic, increasing

function of T, the temperature dependence can be brought inside the gradient operator. This helps in solving equation (7–3.1) conservatively and implicitly.

Equation (7–3.1) has an interesting and instructive analytical solution for a particular propagating profile that has a fixed shape. We assume the temperature profile can be expressed as $T(x - Vt)$, where V is a constant velocity. The time derivative of T in equation (7–3.1) is then

$$\frac{\partial T}{\partial t} = -V\frac{\partial T}{\partial x}. \tag{7–3.2}$$

This leaves a second-order ordinary differential equation in x to be solved for the profile,

$$\frac{dT}{dx} = -\frac{2}{3}\frac{\lambda_{\frac{1}{2}}}{V}\frac{d^2T^{\frac{3}{2}}}{dx^2}. \tag{7–3.3}$$

The first integral of this equation introduces a constant of integration. Equating this constant with the temperature T_∞ far upstream of the advancing nonlinear heat wave gives

$$T(x) - T_\infty = -\frac{2}{3}\frac{\lambda_{\frac{1}{2}}}{V}\frac{dT^{\frac{3}{2}}}{dx}. \tag{7–3.4}$$

Equation (7–3.4) can be integrated analytically to obtain

$$x - x_1 = \int_{x_1}^{x} dx = -\frac{2}{3}\frac{\lambda_{\frac{1}{2}}}{V}\int_{T_1}^{T(x)}\frac{dT^{\frac{3}{2}}}{(T - T_\infty)}, \tag{7–3.5}$$

where T_1 is the temperature at x_1. The integral on the right side of equation (7–3.5) yields

$$x - x_1 = \frac{\lambda_{\frac{1}{2}}}{V}\left[-2\left(\sqrt{T} - \sqrt{T_1}\right) + \sqrt{T_\infty}\ \ln\left|\frac{\sqrt{T} + \sqrt{T_\infty}}{\sqrt{T_1} + \sqrt{T_\infty}}\right|\left|\frac{\sqrt{T_o} - \sqrt{T_\infty}}{\sqrt{T} - \sqrt{T_\infty}}\right|\right]. \tag{7–3.6}$$

Figure 7.5 shows the solution to equation (7–3.1) given by equation (7–3.6) with $V = 5$ cm/s at an initial time $t_o = 0$ and several subsequent times. This solution is a profile with fixed shape that propagates to the right. The effect of the nonlinearity in the diffusion coefficient is large when the medium into which the heat is being conducted is cold, that is, when T_∞ is small. The gradient of temperature is larger at high temperature and smaller at low temperature. This keeps the heat flux relatively constant except where the material is heating up most rapidly. This solution assumes no motion of the fluid in response to the changing temperature so the problem is unrealistic for a gas. It could, however, represent the propagation of a thermal wave in a solid over a limited temperature range, excluding phase changes.

The integral equation analogous to equation (7–3.4) can be integrated analytically for a wide range of nonlinear thermal conductivities with the general power-law form,

$$\lambda(T) \equiv \lambda_\alpha T^\alpha. \tag{7–3.7}$$

In the limiting case where the heat is diffusing into a cold medium with $T_\infty = 0$, the

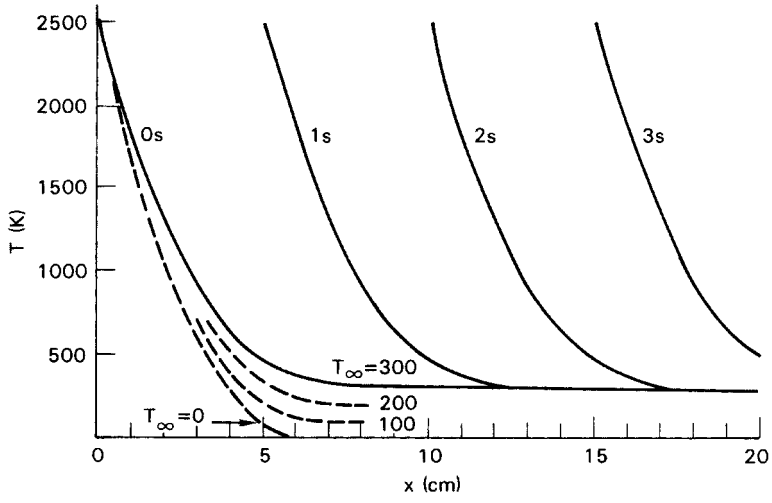

Figure 7.5. Temperature as a function of position for the solution of the thermal conduction front given in equation (7–3.6), with $V = 5$ cm/s. The solid lines are the solution for $T_\infty = 300$ K, given at several times. The dashed lines represent the solution at $t = 0$ for $T_\infty = 0$, 100, and 200 K.

solution is

$$T(x) = \left[\frac{\alpha V}{\lambda_\alpha}(x_1 - x) + T_1^\alpha\right]^{1/\alpha}. \tag{7–3.8}$$

This solution reaches

$$\begin{aligned} &T_1 \quad \text{at} \quad x_1 \quad \text{and passes through} \\ &T = 0 \quad \text{at} \quad x_o = x_1 + \frac{\lambda_\alpha T_1^\alpha}{\alpha V}, \end{aligned} \tag{7–3.9}$$

a finite distance away. In the vicinity of x_o the temperature approaches zero as $(x - x_o)^{1/\alpha}$. When α is less than unity, that is, $\alpha = \frac{1}{2}$ in equation (7–3.6), the leading edge of the temperature in this limiting solution goes smoothly to zero at finite distance with zero slope. The solution is not singular and has infinite slope only as T approaches infinity.

When α is greater than unity, the slope of the front becomes infinite at $x = x_o$ and the thermal conduction wave advances like a shock. The temperature behind the front must have a very steep gradient because the relatively cold medium ahead of the advancing wave has a relatively low conductivity. Such highly nonlinear diffusion equations arise in some models of turbulent transport and turbulent mixing, where they represent zero net convection in regions of the fluid where the vorticity is zero.

The finite temperature of the initial medium keeps the gradient from becoming infinite and causing the diffusion equation to break down. When $T_\infty > 0$, the singular derivative at x_o is replaced by a very thin transition region in which the temperature profile goes smoothly to zero with an exponentially small foot extending to infinity. This behavior applies to the strongly nonlinear case. When $\alpha = 2$, the solution equivalent to equation (7–3.6) is

$$x - x_1 = -\frac{\lambda_2}{V}\left[\frac{T^2 - T_1^2}{2} + T_\infty(T - T_1) + T_\infty^2 \ln\left(\frac{T - T_\infty}{T_1 - T_\infty}\right)\right]. \tag{7–3.10}$$

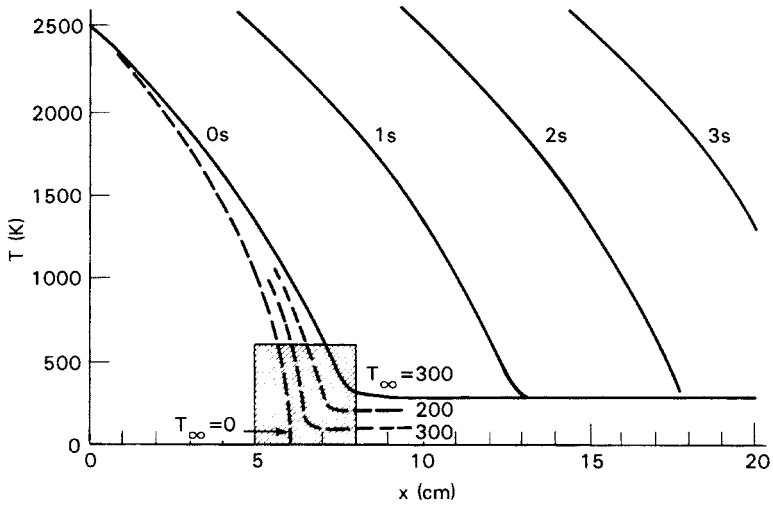

Figure 7.6. Temperature as a function of position for the solution of the thermal conduction front given in equation (7–3.10), with $V = 5$ cm/s and $\alpha = 2$. The solid lines are the solution for $T_\infty = 300$ K, given at several times. The dashed lines represent the solution at $t = 0$ for $T_\infty = 0$, 100, and 200 K. The shaded area indicates that region requiring good numerical resolution when the equation is solved numerically.

Figure 7.6 shows equation (7–3.10) for $V = 5$ cm/s with $x_1 = 0$, $T_1 = 2500$ K, and λ_2 chosen so that the limiting solution goes to zero at 6 cm. These simple nonlinear conduction solutions are closely related to the temperature profiles found in flames and ablation layers.

This analytic solution provides an ideal test of a diffusion algorithm because the correct temperature profile and propagation velocity are known. If the numerical algorithm is conservative, it will propagate the approximate numerical profile at the correct speed. The shape of the profile near T_∞ is generally most sensitive to the algorithms and the grid used.

When the thermal conductivity is highly nonlinear, for example, when $\alpha > 1$, and $T_\infty = 0$, the gradient becomes too large to keep the heat flow roughly constant. The rate at which the shock-like thermal conduction wave advances is determined by the rate at which heat is added at the boundary, in this simplified case by the value of λ_α. As shown in Figures 7.5 and 7.6, the solution as T decreases toward T_∞ has much greater curvature when the nonlinearity is strong ($\alpha > 1$) than when it is weak ($\alpha < 1$). Thus the case $\alpha > 1$ is a more stringent test of nonlinear thermal conduction algorithms. Adequate spatial resolution at the inflection point of the solutions with finite T_∞ is particularly important in obtaining accurate numerical solutions. This region is shaded in Figure 7.6.

7–3.2. Numerical Techniques for Nonlinear Diffusion

Discretizing the partial differential equation describing diffusion can be complicated by the functional form of the diffusion coefficients. Nonetheless, if the timestep is small enough, the changes in cell values $\{\rho_j\}$ are small and the nonlinear terms in the diffusion coefficients do not change fast enough to present major difficulty. Then the nonlinear terms can be evaluated using known quantities at the old time t^{n-1} and errors are small. One or two

iterations, using progressively better estimates of the new values of the variable, are usually needed to reevaluate the nonlinear coefficients and improve the accuracy appreciably. When the timestep is large, such a simple iteration does not necessarily converge. When it does, it may converge very slowly. At even longer timesteps, this prescription is unstable and the computed solution can change appreciably in a single timestep.

When the diffusion problem is highly nonlinear, it is often necessary to use implicit algorithms. Using the general power-law thermal conductivity as an example, the fully implicit three-point formula for the temperature at the new time is

$$\frac{T_i^n - T_i^{n-1}}{\Delta t} = \frac{\lambda_\alpha}{\alpha+1}\left[\frac{T_{i+1}^{\alpha+1} - T_i^{\alpha+1}}{\Delta x^2} - \frac{T_i^{\alpha+1} - T_{i-1}^{\alpha+1}}{\Delta x^2}\right]^n, \tag{7–3.11}$$

where the entire term in brackets on the right side is evaluated at the new time. For this equation, the straightforward use of a tridiagonal solver to find the new temperatures $\{T_i^n\}$ is complicated by the nonlinear terms on the right side.

A quadratically convergent iteration can be used to solve equation (7–3.11). A sequence of solutions $T_i^{(m)}$, where m indicates the iteration number, results from replacing the current best estimates $\{T_i^{(m)}\}$ of the desired new values, $\{T_i^n\}$, with

$$T_i^{(m+1)} = T_i^{(m)} + \Delta T_i^{(m)}, \qquad i = 1, \ldots, N_c. \tag{7–3.12}$$

Equation (7–3.11), with equation (7–3.12) substituted for the desired T_i^n values, yields

$$\begin{aligned}\frac{\Delta T_i^{(m)}}{\Delta t} - \frac{\lambda_\alpha}{\Delta x^2}\left[\left(T_{i+1}^{(m)}\right)^\alpha \Delta T_{i+1}^{(m)} - 2\left(T_i^{(m)}\right)^\alpha \Delta T_i^{(m)} + \left(T_{i-1}^{(m)}\right)^\alpha \Delta T_{i-1}^{(m)}\right] \\ = \frac{T_i^o - T_i^{(m)}}{\Delta t} + \frac{\lambda_\alpha}{\Delta x^2(\alpha+1)}\left[\left(T_{i+1}^{(m)}\right)^{\alpha+1} - 2\left(T_i^{(m)}\right)^{\alpha+1} + \left(T_{i-1}^{(m)}\right)^{\alpha+1}\right].\end{aligned} \tag{7–3.13}$$

With this linearization, the new corrections $\Delta T_i^{(m)}$ can be determined using a tridiagonal solver. In this form, each iteration is conservative, and so the overall solution is conservative.

Equation (7–2.10) is a formula for determining the appropriate average values of diffusion coefficients at cell interfaces, given their values at cell centers. Nonlinear diffusion becomes a difficult problem when the coefficients vary greatly from cell to cell, and so it is natural to apply this flux-matching approach. When the diffusion coefficient is integrable, as in equation (7–3.7), or, more generally, when the flux can be written

$$D(\rho)\nabla\rho = \nabla G(\rho), \tag{7–3.14}$$

using flux matching reduces to a linear interpolation of $G(\rho)$ at the cell interface. This occurs because the effective diffusion coefficient is constant once $D(\rho)$ is brought inside the gradient.

Usually the form of real nonlinear diffusion coefficients is not as simple as that assumed in equation (7–3.7). Consider the case in which the thermal-conduction coefficient varies as $\lambda(\rho, T)$. If the T dependence can be brought inside the gradient, flux matching can be applied to the variation of λ with ρ that stays outside the gradient. Even when all of the variation of λ can be brought inside the gradient, as in equation (7–3.14), flux matching

can be used to find the averaged quantities needed to determine cell-interface fluxes, even when the cell sizes vary.

7–3.3. Asymptotic Methods for Fast Diffusion

When the diffusion coefficient is large everywhere or convection is relatively slow, the source and sink terms dominate and equation (7–1.1) becomes

$$\nabla \cdot D(\rho, x, t)\nabla\rho = -S, \tag{7–3.15}$$

an elliptic equation. Essentially the same structure for the solution matrix results if the time derivative is kept. In the limit of large timesteps, a finite-difference approximation for the time derivative is not expensive computationally and increases the diagonal dominance of the resulting sparse matrix equation.

When D is large and S is finite, the gradient of ρ must become small quickly so that the right and left sides of equation (7–3.15) are equal. In the limit of fast diffusion when the sources are negligible, ρ becomes nearly constant. When the source term is important, ρ has small variations around the constant value. The limit of fast thermal conduction is an isothermal system in which the temperature can be determined from global energy conservation.

When the source terms are present but change slowly compared to the rate at which diffusion can move ρ through the system, an elliptic or parabolic equation must be solved. In one dimension the resulting tridiagonal matrix is straightforward to solve. In multidimensions, slow changes in the solution can be used to simplify the solution procedure. The variation of the diffusion coefficient, if it depends on the slowly changing solution, may not need to be updated during a timestep. The nonlinearities may be evaluated at the beginning of the cycle, thus reducing a potentially nonlinear elliptic equation to a linear problem with known but varying coefficients.

7–3.4. Flux Limiters

A flux-limiting formula is an alternate approach to solving the parabolic or elliptic equation discussed above. The idea of flux limiting is to reduce the flux artificially to a value that is not large enough to evacuate any cells in one timestep. This is equivalent to locally decreasing the timestep below the explicit stability limit. Because the diffusion coefficient is presumably large where flux limiters are applied, the diffusion is still fast and gradients in the solution are still small, although not as small as they would be if the full diffusive fluxes were allowed.

Viewed in this way, flux limiters are useful devices for allowing a larger explicit timestep than numerical stability conditions would otherwise permit. If very fine resolution is required somewhere on the grid for reasons other than resolving the diffusion terms, flux limiters help to avoid the diffusive stability condition that would require exceptionally small timesteps. Flux-limiting formulas also arise naturally from the physics as well as the numerics.

In Section 7–1.3 we showed how the diffusion velocity approaches infinity sufficiently far from the center of a spreading Gaussian. If a quantity – heat or material – is diffusing,

the largest average diffusion speed possible is approximately the thermal velocity of the particle distribution at the point of interest. Thus the flux can be at most $v_{th}\rho_i$ for one of the chemical species, or $Nk_BTv_{th}/(\gamma - 1)$ for the energy, where v_{th} is the local thermal velocity. Actually, the limiting flux is smaller because not all of the particles can be random walking in the same direction simultaneously. Practically, the limiting diffusion velocity is at most a few percent of the thermal velocity.

Flux limiting in diffusion is particularly important in nearly collisionless plasmas with steep temperature gradients, and on the scale of the mean free path in gas-dynamic shocks. In the plasma case, the hot particles have a much longer mean free path than the cold particles and thus can carry mass and heat a long way into a cold medium before they thermalize. Because the local gradients are steep, however, the classical continuum treatment predicts a flux that is far too large. In shocks, the hot particles behind the shock carry mass, momentum, and energy forward into the unshocked medium. If the shock is treated as a discontinuous profile, the diffusive flux may again be too large without flux limiting.

7–4. Multidimensional Algorithms

7–4.1. Implicit, Explicit, and Centered Algorithms

The thermal conduction problem is useful for illustrating the generalizations needed to solve multidimensional diffusion problems. We deal with a system in which the fluid velocity is zero, or assume that the thermal-conduction process is being integrated separately from other physical processes. Then the diffusion of heat is described by

$$\frac{\partial}{\partial t}C(T, \mathbf{x}, t)T(\mathbf{x}, t) = \nabla \cdot \lambda(T, x, t)\nabla T + S(T, \mathbf{x}, t), \tag{7–4.1}$$

where $C(T, \mathbf{x}, t)$ is the specific heat. The term $S(T, \mathbf{x}, t)$ represents temperature or energy sources and sinks, and $\lambda(T, x, t)$ is the coefficient of thermal conduction. When the functions C, λ, or S depend on the temperature, as indicated in equation (7–4.1), the problem is nonlinear, but generalization of the techniques in Section 7–3 can extend the algorithms given there.

When λ and C are constant and S is zero, the one-dimensional solution given in Chapter 4 generalizes easily to two dimensions. Let the temperature profile be a constant everywhere, T_{avg}, with an added sinusoidal fluctuation. We expect the solution at later times to have the form

$$T(x, y, t) = T_{\text{avg}} + \Delta T_o e^{ik_x x} e^{ik_y y} e^{-\Gamma t}, \tag{7–4.2}$$

where ΔT_o is the amplitude of the temperature fluctuation at $t = 0$. Substituting equation (7–4.2) into equation (7–4.1) gives

$$\Gamma = \frac{\lambda}{C}\left(k_x^2 + k_y^2\right), \tag{7–4.3}$$

showing that the decay rate of the various modes increases as the square of the wavenumber.

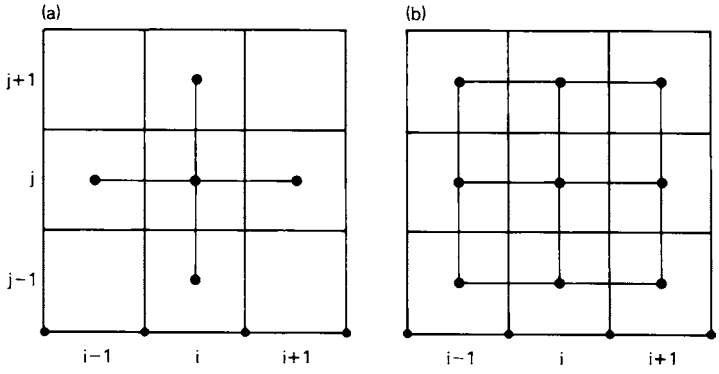

Figure 7.7. Local two-dimensional finite-difference templates. (a) Five-point template. (b) Nine-point template.

Although there are many finite-difference approximations to equation (7–4.1), we adopt a two-dimensional five-point difference template, shown in Figure 7.7a. Then

$$C_{ij}T_{ij}^{n} + \frac{\theta_x \Delta t}{\Delta x_{ij}} F_x^n + \frac{\theta_y \Delta t}{\Delta y_{ij}} F_y^n = C_{ij}T_{ij}^{n-1} + \frac{(1-\theta_x)\Delta t}{\Delta y_{ij}} F_x^{n-1} + \frac{(1-\theta_y)\Delta t}{\Delta y_{ij}} F_y^{n-1}, \tag{7–4.4a}$$

where

$$F_x = \lambda^x_{i+\frac{1}{2},j}\left(\frac{T_{i+1,j} - T_{ij}}{\Delta x_{i+\frac{1}{2},j}}\right) - \lambda^x_{i-\frac{1}{2},j}\left(\frac{T_{ij} - T_{i-1,j}}{\Delta x_{i-\frac{1}{2},j}}\right) \tag{7–4.4b}$$

and

$$F_y = \lambda^y_{i,j+\frac{1}{2}}\left(\frac{T_{i,j+1} - T_{ij}}{\Delta y_{i,j+\frac{1}{2}}}\right) - \lambda^y_{i,j-\frac{1}{2}}\left(\frac{T_{ij} - T_{i,j-1}}{\Delta y_{i,j-\frac{1}{2}}}\right). \tag{7–4.4c}$$

Figure 7.7b shows a nine-point template that can also be used. The extra freedom that exists in nine-point difference formulas can be used to increase the accuracy when cell sizes and diffusion coefficients vary in space.

Equation (7–4.4) ensures conservation of energy when the diffusion coefficient varies across the grid. The grid can be unevenly spaced in both x and y, but the grid lines are assumed to define an orthogonal coordinate system. The cell-centered and interface-centered cell sizes are defined as before,

$$\begin{aligned} \Delta x_{i+\frac{1}{2},j} &\equiv (x_{i+1,j} - x_{i,j}) \\ \Delta x_{ij} &\equiv \left(x_{i+\frac{1}{2},j} - x_{i-\frac{1}{2},j}\right) \end{aligned} \tag{7–4.5a}$$

and

$$\begin{aligned} \Delta y_{i,j+\frac{1}{2}} &\equiv (y_{i,j+1} - y_{ij}) \\ \Delta y_{ij} &\equiv \left(y_{i,j+\frac{1}{2}} - y_{i,j-\frac{1}{2}}\right). \end{aligned} \tag{7–4.5b}$$

The two directions may have different implicitness parameters, θ_x and θ_y, and these can even vary from cell to cell. These equations are a straightforward generalization of equation (7–2.1) and fully second-order accurate when $\theta_x = \theta_y = \frac{1}{2}$ and the cell spacing is uniform in both directions. A three-dimensional finite-difference equation can also be written as an extension of equation (7–4.4).

We now follow basically the same procedure as in Chapter 4: assume that T initially has a sinusoidal spatial distribution consistent with the theoretical solution presented above. Also, we assume that λ is constant in space and time and that the computational cells, Δx and Δy, are uniformly spaced. This produces a two-dimensional analog of equation (4–3.9) that relates the new temperature perturbation to the old one,

$$\Delta T^n = \Delta T^{n-1} \frac{\left\{1 - \frac{\lambda \Delta t}{2C}\left[(1-\theta_x)K_x^2 + (1-\theta_y)K_y^2\right]\right\}}{\left\{1 + \frac{\lambda \Delta t}{2C}\left[\theta_x K_x^2 + \theta_y K_y^2\right]\right\}}. \tag{7–4.6}$$

Here we define

$$K_x^2 = \frac{2(1 - \cos k_x \Delta x)}{(\Delta x)^2} \tag{7–4.7a}$$

$$K_y^2 = \frac{2(1 - \cos k_y \Delta y)}{(\Delta y)^2} \tag{7–4.7b}$$

as the finite-difference approximations to the continuum wavenumbers k_x and k_y. At long wavelengths, where the spatial variations are well resolved, K is essentially the same as k. Equation (7–4.6) is the two-dimensional analog to the one-dimensional amplification factor in equation (4–3.10). The explicit ($\theta = 0$), centered ($\theta = \frac{1}{2}$), and implicit algorithms ($\theta = 1$) behave as they did in the one-dimensional case.

Based on the behavior of the one-dimensional methods and on the two-dimensional numerical amplification factor given by $\Delta T^n / \Delta T^{n-1}$ in equation (7–4.6), the fully implicit algorithm appears to perform the best. In particular, it behaves in a physically stable way at all wavelengths. To solve the multidimensional fast-diffusion problem, it is natural to start off by assuming that the best approach is to generalize the fully implicit algorithm. This is easy to do in principle and the dispersion relation is known, as shown above. Unfortunately, actually solving the implicit system of coupled difference equations in two dimensions is more difficult computationally. If, on the other hand, the implicit system can be solved, the semi-implicit algorithms can also be solved.

Because the finite-difference template couples together values of the temperature implicitly in more than one direction, the matrix that must be inverted is no longer tridiagonal. In two dimensions the direct solution of equation (7–4.4) involves inverting an $N_x N_y \times N_x N_y$ sparse matrix and requires about $N_x^3 \times N_y^3$ operations. This is often too expensive for a reactive-flow calculation. These large sparse matrices are difficult to invert more efficiently than $N_x^3 \times N_y^3$ efficiently, especially in the general case of variable diffusion coefficients and cell sizes. It is necessary either to accept an approximate inverse or an iterative algorithm to find the new temperatures satisfying equation (7–4.4). We refer the reader to Section 10–4 for a more complete discussion of methods for solving the parabolic and elliptic problems.

7–4.2. ADI and Split-Direction Methods

Tridiagonal matrices may be solved quickly and readily by available methods (see Section 10–4). The obvious question is whether a tridiagonal solver can be used to solve the multidimensional problem by alternately applying it to the different spatial directions. This approach leads to the *alternating-direction implicit* or ADI algorithms. Here we briefly outline the application of ADI to the diffusion problem of interest here.

Two steps are required to apply ADI methods to the idealized thermal-conduction problem described above. The first step solves the x-direction derivatives implicitly using the y finite-difference approximations evaluated explicitly at the old time. This step involves solving N_y independent tridiagonal matrix systems, each of length N_x. The next "alternating" step evaluates the x-direction finite-difference approximations explicitly, and solves the y-direction derivatives implicitly. This step involves solving N_x independent tridiagonal solutions. Note that there is a bias in the way the different directions are treated. One semi-implicit way to remove some of the bias is to use the old time values (t^{n-1}), and then average the two intermediate results.

A useful exercise is to write out the equations described for these two approaches, based on equation (7–4.4), and then to derive the dispersion relations for both algorithms. These should then be compared with each other and with the fully implicit dispersion relation that results from solving the sparse matrix system directly.

7–5. Solving for Species Diffusion Velocities

The costs of a reactive-flow calculation are compounded when there are many reacting species. If the cost scaled merely as the number of species at each computational cell, N_s, the problem would be tractable. Whenever a matrix of size $N_s \times N_s$ must be inverted, however, the operation count and hence the computational cost scales as $(N_s)^3$ for each cell.

Calculating the species diffusion velocities, $\{\mathbf{v}_{di}\}$, requires solving equation (7–0.5) and equation (7–0.6), subject to the constraint in equation (7–0.7). We are again faced with the problem of the high computational cost of an expensive but straightforward matrix inversion.

7–5.1. The Fickian Diffusion Approximation

The Fickian diffusion approximation is commonly made to reduce this cost, so understanding its assumptions and limitations is worthwhile. Consider an idealized one-dimensional problem in which temperature and pressure are constant. There are no external forces, no chemical reactions, and no fluid velocity, but the species densities can vary with position. The species continuity equation reduces to

$$\frac{\partial n_i}{\partial t} + \frac{\partial}{\partial x}(n_i v_{di}) = 0, \tag{7–5.1}$$

and a number-averaged velocity, w, can be defined as

$$w \equiv \frac{1}{N}\sum_{i=1}^{N_s} n_i v_{di}, \tag{7–5.2}$$

where N is the total number density, $\sum_{i=1}^{N_s} n_i \equiv N$. Then

$$\frac{\partial w}{\partial x} = 0, \tag{7–5.3}$$

which can be found by summing equation (7–5.1) over all i and noting that $\partial N/\partial t = 0$ is consistent with the assumptions given above. Using equation (7–5.3), we see that

$$\frac{\partial (n_i w)}{\partial x} = w \frac{\partial n_i}{\partial x}. \tag{7–5.4}$$

We can rewrite equation (7–5.1) as

$$\frac{\partial n_i}{\partial t} = -\frac{\partial n_i v_{di}}{\partial x} + \frac{\partial w n_i}{\partial x} - w \frac{\partial n_i}{\partial x}. \tag{7–5.5}$$

Using the definition of w from equation (7–5.2), this becomes

$$\frac{\partial n_i}{\partial t} = -\frac{\partial}{\partial x}\left[n_i \sum_{j=1}^{N_s} \left(\delta_{ij} - \frac{n_j}{N} \right) v_{di} \right] - w(t) \frac{\partial n_i}{\partial x}. \tag{7–5.6}$$

Given the approximations stated at the beginning of this section, the diffusion velocity can be related to the binary diffusion coefficient D_{ij} through the expression for the mass flux (see Hirschfelder, Curtiss, and Bird [1954], hereafter referred to as HCB [1954]),

$$j_i = n_i m_i v_{di} = \frac{N^2}{\rho} \sum_{j=1}^{N_s} m_i m_j D_{ij} \frac{\partial}{\partial x} \left(\frac{n_i}{N} \right). \tag{7–5.7}$$

The final equation combining equations (7–5.6) and (7–5.7) is

$$\frac{\partial n_i}{\partial t} = -\frac{\partial}{\partial x} \sum_{k=1}^{N_s} \sum_{j=1}^{N_s} \left(\delta_{ij} - \frac{n_j}{N} \right) \frac{n_i N}{n_j \rho} m_k D_{jk} \frac{\partial n_k}{\partial x} - w(t) \frac{\partial n_i}{\partial x}. \tag{7–5.8}$$

Equation (7–5.8) is the *the multicomponent generalization of Fick's second law of diffusion*, and is discussed in more detail in HCB (1954).

In the limit of only two species, n_1 and n_2, equation (7–5.8) reduces to

$$\frac{\partial n_1}{\partial t} = -\frac{\partial}{\partial x} D_{12} \frac{\partial n_1}{\partial x} - w(t) \frac{\partial n_1}{\partial x} \tag{7–5.9a}$$

and

$$\frac{\partial n_2}{\partial t} = -\frac{\partial}{\partial x} D_{12} \frac{\partial n_2}{\partial x} - w(t) \frac{\partial n_2}{\partial x}. \tag{7–5.9b}$$

Summing these gives

$$\frac{\partial (n_1 + n_2)}{\partial t} = \frac{\partial N}{\partial t} = 0, \tag{7–5.10}$$

consistent with the assumptions of constant temperature and pressure made above.

An expression of the form

$$\frac{\partial C}{\partial t} = \frac{\partial}{\partial x} D \frac{\partial C}{\partial x}, \tag{7–5.11}$$

where C is a concentration, is commonly called *Fick's law* or *Fickian diffusion*. Note that the first-order derivative term found in equation (7–5.9) is absent and the assumptions of constant temperature, pressure, and so on, made above may not actually be satisfied in practical applications.

7–5.2. An Efficient Iterative Algorithm

Because the straightforward inversion of equations (7–0.5) through (7–0.7) to find the diffusion velocities requires on the order of N_s^3 arithmetic operations per cell, a more efficient method is needed when there are more than four or five species present. One efficient iterative algorithm was presented by Jones and Boris (1981) (see also Oran and Boris [1981]), and later a rigorous analysis was given by Giovangigli (1991). This algorithm requires of order N_s^2 operations and is based on a special initial guess. To understand the procedure, we first rewrite equations (7–0.5) and (7–0.6) as

$$G_i = \sum_{k=1}^{N_s} W_{ik}(v_{dk} - v_{di}) \tag{7–5.12}$$

and

$$G_i = \nabla\left(\frac{n_i}{N}\right) - X_i \frac{\nabla P}{P} - Y_i \frac{\nabla T}{T}. \tag{7–5.13}$$

Here the $\{W_{ik}\}$ and $\{Y_i\}$ are now functions of the transport coefficients and the vector notation has been dropped as each direction can be treated independently.

When these equations are summed over all species, we find that

$$\sum_{i=1}^{N_s} G_i = 0. \tag{7–5.14}$$

Because the values of G_i sum to zero, the matrix $\mathbf{W}$ is singular as defined in equations (7–5.12) and (7–0.5), and the N_s different diffusion velocities cannot all be independent. The extra equation needed is the constraint equation (7–0.7), which is written here as

$$\sum_{i=1}^{N_s} \rho_i v_{di} = 0. \tag{7–5.15}$$

If a particular set of solutions $\{v_{di}^p\}$ is known for equation (7–5.12), any constant velocity may be added giving another equally correct solution $\{v_{di}^p + \delta v_d\}$ for $i = 1, \ldots, N_s$. The as yet undetermined constant, δv_d, may be used to enforce the constraint equation (7–5.15).

Let the diffusion coefficient D_{is} describe diffusion of species i through the background provided by the sum of all other species. Then an excellent starting approximation for the

ith diffusion velocity, v_{di}^o, is

$$v_{di}^o \equiv -\frac{\rho - \rho_i}{\rho} \frac{N^2 D_{is}}{(N - n_i) n_i} G_i. \tag{7–5.16}$$

Higher-order terms that improve v_{di}^o may be written as δv_{di}, where

$$v_{di} \equiv v_{di}^o + \delta v_{di}. \tag{7–5.17}$$

The equations for $\{\delta v_{di}\}$ are found by substituting equation (7–5.17) into equation (7–5.12), giving

$$\delta G_i \equiv \sum_{k=1}^{M} A_{ik} G_k = \sum_{k=1}^{M} W_{ik} (\delta v_{dk} - \delta v_{di}). \tag{7–5.18}$$

The $\{D_{is}\}$ in equation (7–5.16) are defined as

$$\frac{D_{is}}{(N - n_i)} \sum_{k \neq i} \frac{n_k}{D_{ik}} \equiv 1, \tag{7–5.19}$$

and the matrix elements of $\mathbf{A}$ are given by

$$A_{ik} \equiv \frac{\rho_i}{\rho} \delta_{ik} + \frac{n_i}{D_{ik}} \frac{(\rho - \rho_k)}{\rho} \frac{D_{ks}}{(N - n_k)} (1 - \delta_{ik}). \tag{7–5.20}$$

Equation (7–5.18) defines a linear system of equations that can be solved for the $\{\delta v_{di}\}$ and this is basically the same equation we started with: equation (7–5.12)! The right side of equation (7–5.18) vanishes when summed over i. It is easy to see that the choice of A_{ik} here is not unique. Each row of $\mathbf{A}$ can have an arbitrary constant added according to

$$\tilde{A}_{ik} \equiv A_{ik} - C_i \tag{7–5.21}$$

without changing equation (7–5.18), which is consistent with equation (7–5.15). Such constants leave $\{\delta v_{di}\}$ unchanged. The general form of the complete solution is, therefore,

$$v_{di} = \frac{-(\rho - \rho_i)}{\rho} \frac{N^2 D_{is}}{(N - n_i) n_i} \times [\delta_{ik} + A_{ik} + A_{il} A_{lk} + \cdots] G_k. \tag{7–5.22}$$

The matrix in square brackets is the formal expansion of $[1 - \mathbf{A}]^{-1}$.

In the numerical evaluation of equation (7–5.22), none of the indicated matrix multiplications have to be performed. Since $\{G_i\}$ is known, multiply from the right first. Each additional power of A is obtained by multiplying a vector, rather than a matrix, by A, giving a computational cost of order N_s^2. In practice, it is convenient to take $C_i = 0$ in equation (7–5.21), and to truncate the expansion in equation (7–5.22) at the A^2 term. At least the first correction A_{ik} needs to be included to get the correct sign for all the diffusion fluxes. The quadratic term sometimes adds significant extra accuracy and further iteration is generally unnecessary. The errors remaining are at most a few percent.

This algorithm converges quickly because of the initial approximation embodied in equations (7–5.16) and (7–5.19). The factor $(\rho - \rho_i)/\rho$ in equation (7–5.22) is crucial. When there are only two species, so that $N = n_1 + n_2$, this factor becomes ρ_2/ρ as required

to give the exact two species result. Note that terminating the expansion of equation (7–5.22) at the δ_{ik} term does not give Fickian diffusion. The effective diffusion coefficient differs from Fickian by the factor $(\rho - \rho_i)/\rho$. Fickian diffusion may not even give the correct sign for the diffusive flux.

7–6. Evaluating Diffusive Transport Coefficients

The purpose of this section is to provide some useful formulas and references for evaluating transport coefficients in neutral gases. Much of the fundamental and extensive work in this area was done by Chapman and Cowling (1952) and HCB (1954). The material presented below is based on, extends, or applies this work. In addition, a recent book by Ern and Giovangigli (1994) gives a rigorous mathematical derivation that extends the discussion to solution algorithms.

7–6.1. Thermal Conductivity

For multicomponent reactive flows, we need the value of the coefficient of thermal conductivity appropriate for a mixture of neutral gases, λ_m, and λ_i, the coefficient for a pure gas species, i.

For the thermal conductivity of the mixture, an extremely useful equation has been given by Mason and Saxena (1958),

$$\lambda_m = \sum_i \lambda_i \left[1 + \frac{1.065}{2\sqrt{2} n_i} \sum_{k \neq i} n_k \phi_{ik} \right]^{-1}, \tag{7–6.1}$$

where ϕ_{ik} is given by

$$\phi_{ik} = \frac{\left[1 + \left(\lambda_i^0/\lambda_k^0\right)^{1/2} (m_i/m_k)^{1/4}\right]^2}{[1 + m_i/m_k]^{1/2}}. \tag{7–6.2}$$

If species i is a monotonic gas, λ_i^0 is the thermal conductivity of species i. If i is a polyatomic gas, λ_i^0 is the thermal conductivity with the internal degrees of freedom considered "frozen." The $\{m_i\}$ are the atomic masses of the different species.

The thermal conductivity of a pure polyatomic gas, λ_i, can be estimated from the λ_i^0 by using the *Euken factor*, E_i (Hirschfelder 1957b),

$$\lambda_i = E_i \lambda_i^0, \tag{7–6.3}$$

where

$$E_i = 0.115 + 0.354 \frac{c_{pi}}{k_B}, \tag{7–6.4}$$

and c_{pi} is the specific heat at constant pressure. The $\{\lambda_i^0\}$ may be evaluated from

$$\lambda_i^0 = \frac{8.322 \times 10^3}{\sigma_i^2 \Omega_{ii}^{(2,2)*}} \left(\frac{T}{m_i} \right)^{1/2}. \tag{7–6.5}$$

Here σ_i is the *collision diameter*. The quantity $\Omega_{ii}^{(2,2)*}$ is a *collision integral* normalized to its rigid sphere value. It is a function of T_i^*, the reduced temperature

$$T_i^* = \frac{k_B T}{\epsilon_i}, \tag{7–6.6}$$

where ϵ_i / k_B is the *potential parameter*, and k_B is Boltzmann's constant. The ϵ_i and the σ_i are constants in the intermolecular potential function describing the interaction between two molecules of type i. In this formulation, the collision integral has been evaluated using a Lennard-Jones potential between molecules. In general, tables of the $\Omega_{ii}^{(2,2)*}$ for a Lennard-Jones 6–12 potential can be found in HCB (1954) and for more general Lennard-Jones potentials in Klein et al. (1974).

The expression given for $\{\lambda_i^0\}$ is excellent for nonpolar molecules and probably adequate for polar molecules. The origin of the uncertainty lies in the evaluation of the Euken factor, which is an average over microscopic states. The derivation of E_i assumes that all states have the same diffusion coefficients and does not allow for distortions in the electron density distribution function due to rotational transitions. These may be significant even at room temperature for polar molecules (Hirschfelder 1957a,b).

7–6.2. Ordinary (or Molecular or Binary) Diffusion

The quantity D_{ik} represents the coefficient for ordinary diffusion of a pair of species (i, k) when there are only two species present. Although not strictly equal to the diffusion coefficient of (i, k) in a mixture with many species present, we assume that they are equal to the accuracy needed. For further discussion of the differences between binary and multicomponent diffusion coefficients, see HCB (1954).

For some binary mixtures of dilute gases, Mason and Marrero (1972) give a semiempirical expression that describes the variation of D_{ik} over a range of temperatures,

$$D_{ik} = \frac{A_{ik} T^{B_{ik}}}{N}. \tag{7–6.7}$$

When the coefficients A_{ik} and B_{ik} are not available, the D_{ik} may be estimated from

$$D_{ik} = \frac{2.628 \times 10^{-3}}{P \sigma_{ik}^2 \Omega_{ik}^{(1,1)*}} \left[\frac{T^3 (m_i + m_k)}{2 m_i m_k} \right]^{1/2}. \tag{7–6.8}$$

The $\Omega_{ik}^{(1,1)*}$ is a collision integral normalized to its rigid sphere value, which is a function of the reduced temperature

$$T_{ik}^* = \frac{k_B T}{\epsilon_{ik}}. \tag{7–6.9}$$

Again collision diameters and potential parameters are needed, this time between two different types of molecules, i and k. In most cases, the values of σ_{ik} and ϵ_{ik} are not available and are usually estimated from

$$\sigma_{ik} = \frac{1}{2}(\sigma_i + \sigma_k) \tag{7–6.10}$$

and

$$\epsilon_{ik} = (\epsilon_i \epsilon_k)^{1/2}. \tag{7–6.11}$$

The collision integral used here again assumes a Lennard-Jones potential, and tables are given in HCB (1954) and Klein et al. (1974).

7–6.3. Thermal Diffusion

Thermal diffusion is a second-order effect which is only important when there are large differences between the atomic masses of the constituent species. The thermal diffusion ratio K_i^T is a measure of the relative importance of thermal to ordinary diffusion. It can be written as (Chapman and Cowling 1952)

$$K_i^T = \frac{1}{5k_B N^2} \sum_k \frac{\left(6C_{ik}^* - 5\right)}{D_{ik}} \left[\frac{n_i m_i a_k - n_k m_k a_i}{m_i + m_k} \right]. \tag{7–6.12}$$

The quantity a_i is the contribution of the ith species to the thermal conductivity of the mixture (assuming that the internal degrees of freedom are frozen):

$$a_i = \lambda_i^0 \left[1 + \frac{1.065}{2\sqrt{2} n_i} \sum_{k \neq i} n_k \phi_{ik} \right]^{-1}. \tag{7–6.13}$$

In equation (7–6.13), ϕ_{ik} is given in equation (7–6.2) and C_{ik}^* is a ratio of collision integrals,

$$C_{ik}^* = \frac{\Omega_{ik}^{(1,2)*}}{\Omega_{ik}^{(1,1)*}}. \tag{7–6.14}$$

The expression for K_i^T has been derived by assuming that the gases in the mixture are monatomic or that the internal degrees of freedom are frozen. Chapman and Cowling (1952) conjectured that the internal degrees of freedom of the gas molecules have a smaller effect on the thermal diffusion ratio than on the thermal conductivity.

7–6.4. Viscosity

Two viscosity coefficients, the bulk viscosity κ and the shear viscosity μ, appear in equation (7–0.8). The bulk viscosity is generally small, and we ignore it throughout this text. The shear viscosity coefficient is more important. In gases and gas mixtures, the shear viscosity coefficient is also relatively small. Its most important effect is to produce the boundary layer near the interface of two materials. In many simulations, the diffusive effects of viscosity are swamped by numerical diffusion. In order to calculate boundary layers from first principles, it is necessary to resolve the effects of physical viscosity near the interface or boundary.

The most convenient form for the viscosity of a gas mixture has been given by Wilke (1950),

$$\mu_m = \sum_i \mu_i \left[1 + \frac{\sqrt{2}}{4n_i} \sum_{k \neq i} n_k \phi'_{ik} \right]^{-1}. \tag{7–6.15}$$

Here the summations are over species and μ_i is the viscosity of the pure component i. To first order,

$$\phi'_{ik} = \frac{[1 + (\mu_i/\mu_k)^{1/2}(m_k/m_i)^{1/4}]^2}{[1 + (m_i/m_k)]^{1/2}}. \tag{7–6.16}$$

The individual μ_i may be determined approximately from (HCB 1954, Chapter 8)

$$\mu_i = \frac{2.67 \times 10^{-5}\sqrt{m_i T}}{\sigma_i^2 \Omega^{(2,2)*}}, \tag{7–6.17}$$

(where μ_i has units of g/cm-s in cgs units). If $\Omega^{(2,2)*} \approx 1$, equation (7–6.16) can be reduced to

$$\phi'_{ik} = \frac{[1 + \sigma_k/\sigma_i]^2}{[1 + (m_i/m_k)]^{1/2}}. \tag{7–6.18}$$

Equation (7–6.15) differs from the form given in HCB in that pure component viscosity coefficients are required instead of binary diffusion coefficients.

7–6.5. Input Data for Diffusive-Transport Coefficients

The main requirement for determining suitable values of diffusion coefficients is finding collision integrals Ω, collision diameters σ, potential parameters ϵ, and so on. Estimates of these parameters and the assumptions on which they are evaluated (for example, the type of intermolecular potential assumed) can be found in many of the texts referenced above. These are also given in the report by Svehla (1962), the book by Hirshfelder et al. (1954), and many other texts, such as Ern and Giovangigli (1994).

In addition, there are a number of available codes that compute these quantities. These are given in the same spirit as the packages for chemical-reaction rate mechanisms, as discussed in Chapter 5. In particular, we recommend the program by Kee et al. (1996), which uses a first-order iteration, and the modified Fickian form (Coffee and Heimerl 1981), to evaluate diffusion velocities and various rules similar to those given earlier for evaluating the various diffusion coefficients. As a rule of thumb for normal combustion problems, similar chemical species have similar or appropropriately scaled values of these parameters, and the overall solution may be relatively insensitive to small variations in the parameters.

REFERENCES

Chapman, S., and T.G. Cowling. 1952. *The mathematical theory of nonuniform gases*. Cambridge, England: Cambridge University Press.

Coffee, T.P., and Heimerl, J.M. 1981. Transport algorithms for premixed laminar steady-state flames. *Combustion and Flame* 43:273–289.

Ern, A., and V. Giovangigli. 1994. *Multicomponent transport algorithms*. Berlin: Springer-Verlag.

Giovangigli, V. 1991. Convergent iterative methods for multicomponent diffusion. *Impact of Computing in Science and Engineering* 3:244–276.

Hirschfelder, J.O. 1957a. Heat conductivity in polyatomic electronically excited or chemically reacting mixtures. In *Sixth Symposium (International) on Combustion*, 351–366. Pittsburgh: The Combustion Institute.

———. 1957b. Heat transfer in chemically reacting mixtures, I. *J. Chem. Phys.* 26:274–282.

Hirschfelder, J.O., C.F. Curtiss, and R.B. Bird. 1954. *Molecular theory of gases and liquids*. New York: John Wiley and Sons.

Jones, W.W., and J.P. Boris. 1981. An algorithm for multispecies diffusion fluxes. *Comp. Chem.* 5: 139–146.

Kee, R.J., G. Dixon-Lewis, J. Warnatz, M.E. Coltrin, and J.A. Miller. 1996. *A Fortran computer code package for the evaluation of gas-phase multicomponent transport properties*. Sandia report SAND86-8246-UC-401, Livermore, Calif.: Sandia National Laboratories.

Klein, M., H.J.M. Hanley, F.J. Smith, and P. Holland. 1974. *Tables of collision integrals and second virial coefficients for the (m,6,8) intermolecular potential function*. National Standard Reference Data Series, No. 47. Gaithersburg, Md.: National Bureau of Standards.

Mason, E.A., and T.R. Marrero. 1972. Gaseous diffusion coefficients. *J. Phys. Chem. Reference Data* 1:3–118.

Mason, E.A., and S.C. Saxena. 1958. Approximate formula for the thermal conductivity of gas mixtures. *Phys. Fluids* 1:361–369.

Oran, E.S., and J.P. Boris. 1981. Detailed modelling of combustion processes. *Progress in Energy and Combustion Sciences* 7:1–72.

Svehla, R.A. 1962. *Estimated viscosities and thermal conductivities of gases at high temperatures*. Washington, D.C.: NASA, Technical Report No. R-132.

Wilke, C.R. 1950. A viscosity equation for gas mixtures. *J. Chem. Phys.* 18:517–519.

8

Computational Fluid Dynamics: Continuity Equations

Solving a system of coupled, multidimensional continuity equations,

$$\frac{\partial \boldsymbol{\rho}}{\partial t} + \nabla \cdot \mathbf{f}(\boldsymbol{\rho}) = 0, \tag{8–0.1}$$

is a fundamental part of simulating reactive flows. As written in equation (8–0.1), $\boldsymbol{\rho}$ could be a single convected scalar, such as a density, a vector of scalars $\{\rho_i\}$, or even the primary variables in the fluid dynamics equations, $(\rho, \rho\mathbf{v}, E)$. This kind of equation is often called a *conservation law* or a *system of conservation laws.*

Chapter 2 introduced fluid convection, and Chapter 4 described and analyzed a number of relatively simple methods for solving a one-dimensional form of this equation,

$$\frac{\partial \rho}{\partial t} + \frac{\partial}{\partial x}(\rho v) = 0, \tag{8–0.2}$$

where $\mathbf{f}$ and $\boldsymbol{\rho}$ in equation (8–0.1) are scalars,

$$f = \rho v. \tag{8–0.3}$$

The standard test problem was a square wave in $\rho(x, t)$ moving at constant velocity v. Equation (8–0.2) was solved for this initial condition using five different finite-difference methods, as shown in Figure 4.6. Although the first three methods, a simple explicit, a centered, and a fully implicit algorithm, may seem *a priori* to be reasonable approaches to solving the problem, they give very poor results. Large-amplitude errors and phase errors rapidly degrade the solutions until they became unusable. The fourth method, the first-order donor-cell method, is better, but its major drawback is that it is too diffusive. The Lax-Wendroff algorithm is less diffusive than the donor-cell algorithm, but the solution exposes the phase errors that introduce unphysical oscillations. Such phase errors are also an intrinsic part of higher-order methods.

This chapter first deals with the solution of a scalar one-dimensional or multidimensional continuity equation with the generic form slightly simplified from equation (8–0.1),

$$\frac{\partial \rho}{\partial t} + \nabla \cdot \rho\mathbf{v} = 0. \tag{8–0.4}$$

The discussion is then extended to coupled systems of continuity equations. We concentrate on explicit Eulerian methods that are generally stable if the Courant-Friedrichs-Lewy (CFL) condition (Courant, Friedrichs, and Lewy 1928) is used to limit the timestep Δt by requiring

$$\Delta t < \frac{\Delta x}{|\mathbf{v}|}, \tag{8–0.5}$$

where $|\mathbf{v}|$ is the largest characteristic velocity in the system and Δx is the computational cell size. Explicit methods are commonly used, easy to program, and the most robust and accurate. Confining the discussion to explicit Eulerian approaches separates the complexities of the solution algorithm from those of the structure of the computational grid and the additional physical effects that occur in, for example, solutions of the Navier-Stokes equations. More complex solutions and problems are the topic of Chapter 9.

Solving equation (8–0.1) or (8–0.4) has been the focus of substantial efforts in the last thirty-five years. The result of this work has been a series of concepts and implementations with important underlying similarities and differences. The major concepts can be divided into two categories: *flux limiting* and *upwinding*. There are many implementations of each of these. The interplay and relation between these two concepts are discussed briefly by Zalesak (1997). For physically motivated introductions to these methods, we recommend the discussions in Anderson (1995), Tannehill, Anderson, and Pletcher (1997), and Laney (1998). For a more advanced, mathematical summary and classification of the major issues in the field of solvers for conservation equations, we recommend LeVeque (1992). Detailed discussion of flux limiting and upwinding can be found in Fletcher (1988) and Hirsch (1988, 1990), which we also recommend generally as excellent, comprehensive books on computational fluid dynamics (CFD).

The first concept, flux limiting, is based on adjusting the fluxes going in and out of computational cells. The objective is to ensure that no unphysical local minima or maxima are introduced by the numerical convection algorithm. Flux limiting has the effect of imposing the fundamental monotonicity and causality properties of an individual continuity equation directly on the numerical solutions. It is basic to a class of nonlinear monotone (positivity-preserving) algorithms such as FCT (flux-corrected transport), PPM (piecewise-parabolic method), HOG (higher-order Gudonov), and TVD (total-variation diminishing). Flux limiting arises naturally in discretizing partial differential continuity equations while attempting to reproduce their important physical and mathematical properties. Thus it is one of the principal subjects of this chapter.

The second concept, upwinding, is based on imposing additional physical constraints on the numerical solutions. The origin of these constraints is in the interactions that occur in coupled sets of continuity equations, such as the Euler or Navier-Stokes equations. Upwinding algorithms try to ensure that "numerical information" about the solution is passed only in the "correct" directions on the computational grid, as determined by the physical characteristics of the flow. These ideas are the basis for the Godunov methods. The result is that greater accuracy is possible for particular problems at the cost of some restrictions in the generality of the approach. This is discussed further in Chapter 9.

This chapter attempts to explain and organize years of work on methods for adequately solving the continuity equation numerically. We deal first with the solution of a scalar continuity equation and then extend the discussion to include sets of coupled continuity

equations. We begin with a brief description of commonly used, classical methods and use these to introduce the concepts of monotonicity, positivity, and nonlinear flux limiting. An analysis of one particular flux-limiting algorithm, FCT, shows how fundamental physical constraints can be incorporated into a solution algorithm. Spectral and finite-elements methods use larger grid-point *stencils*, or *templates*, than are typically used in explicit finite-difference and finite-volume computation. Although these methods seldom incorporate monotonicity constraints, they have strengths that more recent approaches, such as spectral elements and wavelets, attempt to exploit. The chapter is concluded with a discussion of the limitations on resolution and how these relate to the Reynolds number of the flow.

Chapter 9 continues the topics in this chapter by focusing on CFD methods that incorporate additional flow physics intended to further improve the quality of the solutions. One way this is done is to include more specific physical constraints in the solutions. The additional price paid is greater computer time and memory and algorithms that are less general.

8–1. More on Finite-Difference Methods for Convection

Better convection algorithms exist than the simplest methods introduced in Chapter 4. Consider first a series of important algorithms that are the basis for the more advanced classes of algorithms described in subsequent sections. One class of conservative finite-difference methods, sometimes called Lax-Wendroff methods (Lax and Wendroff 1960, 1964), consist of a one-step method and a series of two-step methods. These methods have played a pivotal role in the development of CFD. One of these, the MacCormack method (MacCormack 1969, 1971, 1981), has been a mainstay of computational aerodynamics. Although these relatively straightforward, efficient methods can be used with some confidence when the convected functions are fairly smooth, they have problems when there are steep gradients in the flow. They then need extra, usually unphysical diffusion to avoid large oscillations and numerical instability. Additional references for these algorithms include Richtmyer and Morton (1967) and Roache (1982). Padé or compact finite-difference methods are also discussed at the end of this section. Although these implicit methods may be high order, they are expensive and have the same tendency to produce unphysical oscillations in the presence of steep gradients. An overview and history of these algorithms are given by Hirsch (1990).

8–1.1. The Basic One-Step Method

The class of Lax-Wendroff methods, one of which was used in Chapter 4 to solve the moving-step or square-wave problem, is based on a Taylor-series expansion of the variable ρ in time,

$$\rho(\mathbf{x}, t^{n+1}) = \rho(\mathbf{x}, t^n) + \Delta t \frac{\partial}{\partial t}\rho(\mathbf{x}, t^n) + \frac{1}{2}\Delta t^2 \frac{\partial^2}{\partial t^2}\rho(\mathbf{x}, t^n) + \mathcal{O}(\Delta t^3). \tag{8–1.1}$$

We now change notation to write the expansion for the scalar density ρ as

$$\rho^{n+1}(\mathbf{x}) = \rho^n(\mathbf{x}) + \Delta t \frac{\partial}{\partial t}\rho^n(\mathbf{x}) + \frac{1}{2}\Delta t^2 \frac{\partial^2}{\partial t^2}\rho^n(\mathbf{x}) + \mathcal{O}(\Delta t^3), \tag{8–1.2}$$

where $\rho^{n+1}(\mathbf{x})$ stands for $\rho(\mathbf{x}, t^{n+1})$. The next step is to express $\partial\rho/\partial t$ and $\partial^2\rho/\partial t^2$ in terms of spatial gradients using equation (8–0.5).

In one dimension,

$$\frac{\partial \rho}{\partial t} = -\frac{\partial}{\partial x} f(\rho). \tag{8–1.3}$$

Then

$$\frac{\partial^2 \rho}{\partial t^2} = \frac{\partial}{\partial t}\left(-\frac{\partial f}{\partial x}\right) = -\frac{\partial}{\partial x}\frac{\partial f}{\partial t} = -\frac{\partial}{\partial x}\left(-A\frac{\partial f}{\partial x}\right) = \frac{\partial}{\partial x} A \frac{\partial f}{\partial x}, \tag{8–1.4}$$

where

$$\frac{\partial f}{\partial t} = -A\frac{\partial f}{\partial x}, \tag{8–1.5}$$

using

$$A \equiv \frac{\partial f}{\partial \rho}. \tag{8–1.6}$$

For example, in the density continuity equation (8–0.2),

$$\begin{aligned} f &= \rho v, \\ A &= v, \\ \frac{\partial \rho}{\partial t} &= -\frac{(\partial \rho v)}{\partial x}, \\ \frac{\partial^2 \rho}{\partial t^2} &= \frac{\partial}{\partial x}\left(v\frac{\partial \rho v}{\partial x}\right). \end{aligned} \tag{8–1.7}$$

Now the Lax-Wendroff difference approximation to equation (8–1.2) becomes

$$\rho^{n+1} \cong \rho^n - \Delta t \frac{\partial f}{\partial x} + \frac{1}{2}\Delta t^2 \left(\frac{\partial}{\partial x} A \frac{\partial f}{\partial x}\right), \tag{8–1.8}$$

with

$$\frac{\partial f}{\partial x} \cong \frac{f_{i+1} - f_{i-1}}{2\Delta x} \tag{8–1.9}$$

and

$$\frac{\partial}{\partial x} A \frac{\partial f}{\partial x} \cong \frac{1}{\Delta x}\left(A_{i+\frac{1}{2}} \frac{(f_{i+1} - f_i)}{\Delta x} - A_{i-\frac{1}{2}} \frac{(f_i - f_{i-1})}{\Delta x}\right). \tag{8–1.10}$$

Here we have defined

$$A_{i\pm\frac{1}{2}} \equiv \frac{1}{2}(A_i + A_{i\pm 1}). \tag{8–1.11}$$

The second-derivative term in equation (8–1.8) is an added diffusion term proportional to the square of the velocity when the velocity is constant in space, but it is more complicated when the velocity varies significantly. The role of this term is to correct for the instability of the forward time difference.

In multidimensions, equation (8–1.7) generalizes to

$$\frac{\partial \rho}{\partial t} = -\nabla \cdot \mathbf{f} \tag{8–1.12}$$

and

$$\frac{\partial^2 \rho}{\partial t^2} = \nabla \cdot (\mathbf{A} \cdot (\nabla \mathbf{f})). \tag{8–1.13}$$

8–1.2. Two-Step Richtmyer Methods

As can be seen from the one-dimensional analysis just given above, differencing the two-dimensional equations will be even more complicated using one-step methods. There is an approach to the Lax-Wendroff method, however, which is easy to program in one dimension and extends simply to two and three dimensions.

Richtmyer (1963) showed that the Lax-Wendroff methods can be written as two-step procedures in which the first step is a Lax method to provide provisional values and the second step is a leapfrog method. The two-step Lax-Wendroff method in one dimension is

$$\begin{aligned} \rho_i^{n+1} &= \frac{1}{2}[\rho_{i+1} + \rho_{i-1}]^n - \Delta t \left[\frac{f_{i+1} - f_{i-1}}{2\Delta x}\right]^n, \qquad \text{for odd } i \\ \rho_i^{n+2} &= \rho_i^n - 2\Delta t \left[\frac{f_{i+1} - f_{i-1}}{2\Delta x}\right]^{n+1}, \qquad \text{for even } i, \end{aligned} \tag{8–1.14}$$

where the superscript n indicates that the quantities in brackets are evaluated at the discrete time level t^n. The first step is a provisional "half" step. The values $\{f_{i\pm1}^{n+1}\}$ in the second step, or "whole" step, are evaluated using the quantities $\{\rho_{i\pm1}^{n+1}\}$ found in the *half step*. Substituting the first step into the second step recovers the one-step Lax-Wendroff method for the linear continuity equation with constant velocity v. The more common way of writing equation (8–1.14) is

$$\begin{aligned} \rho_{i+\frac{1}{2}}^{n+\frac{1}{2}} &= \frac{1}{2}[\rho_{i+1} + \rho_i]^n - \frac{\Delta t}{2}\left[\frac{f_{i+1} - f_i}{\Delta x}\right]^n \\ \rho_i^{n+1} &= \rho_i^n - \Delta t \left[\frac{f_{i+\frac{1}{2}} - f_{i-\frac{1}{2}}}{\Delta x}\right]^{n+\frac{1}{2}}. \end{aligned} \tag{8–1.15}$$

In this form, the half-step values $\{\rho_{i+\frac{1}{2}}^{n+\frac{1}{2}}\}$ are evaluated on a staggered grid and then used to evaluate centered derivatives on the original grid to determine $\{\rho_i^{n+1}\}$.

This version of the two-step method may be generalized to multidimensions. For example, in two dimensions,

$$\begin{aligned}
\rho_{i+\frac{1}{2},j}^{n+\frac{1}{2}} &= \frac{1}{2}[\rho_{i,j} + \rho_{i+1,j}]^n \\
&\quad - \frac{\Delta t}{2}\left[\frac{(f_x)_{i+1,j} - (f_x)_{i,j}}{\Delta x} + \frac{(f_y)_{i+\frac{1}{2},j+1} - (f_y)_{i+\frac{1}{2},j-1}}{2\Delta y}\right]^n \\
\rho_{i,j+\frac{1}{2}}^{n+\frac{1}{2}} &= \frac{1}{2}[\rho_{i,j} + \rho_{i,j+1}]^n \\
&\quad - \frac{\Delta t}{2}\left[\frac{(f_x)_{i+1,j+\frac{1}{2}} - (f_x)_{i-1,j+\frac{1}{2}}}{2\Delta x} + \frac{(f_y)_{i,j+1} - (f_y)_{i,j}}{\Delta y}\right]^n \\
\rho_{i,j}^{n+1} &= \rho_{i,j}^n \\
&\quad - \Delta t\left[\frac{(f_x)_{i+\frac{1}{2},j} - (f_x)_{i-\frac{1}{2},j}}{\Delta x} + \frac{(f_y)_{i,j+\frac{1}{2}} - (f_y)_{i,j-\frac{1}{2}}}{\Delta y}\right]^{n+\frac{1}{2}}.
\end{aligned} \tag{8–1.16}$$

The half-step values in the first two equations may be defined, for example, as

$$f_{i\pm\frac{1}{2}}^{n+\frac{1}{2}} = f\left(\rho_{i\pm\frac{1}{2}}^{n+\frac{1}{2}}\right) \equiv f\left(\frac{1}{2}\left(\rho_i^{n+\frac{1}{2}} + \rho_{i\pm1}^{n+\frac{1}{2}}\right)\right), \tag{8–1.17}$$

or as

$$f_{i+\frac{1}{2}}^{n+\frac{1}{2}} = \frac{1}{2}\left[f\left(\rho_i^{n+\frac{1}{2}}\right) + f\left(\rho_{i+1}^{n+\frac{1}{2}}\right)\right], \tag{8–1.18}$$

where f with no subscripts or superscripts indicates "a function of" the quantity in parentheses. Emery (1968) showed that this two-step method is about four times faster than the one-step Lax-Wendroff in two dimensions. There are a number of modifications of the basic two-step method (see, for example, Roache [1982]), including extensions to higher orders of accuracy in space and time (given, for example, by Gottlieb and Turkel [1976] and Zalesak [1984]). A summary and some discussion of these methods is given in Hirsch (1990).

8–1.3. The MacCormack Method

The one-dimensional form of the two-step MacCormack method [MacCormack, 1969] is

$$\begin{aligned}
\rho_i^* &= \rho_i^n - \Delta t\left[\frac{f_{i+1}^n - f_i^n}{\Delta x}\right] \\
\rho_i^{n+1} &= \frac{1}{2}\left\{\rho_i^n + \rho_i^* - \Delta t\left[\frac{f_i^* - f_{i-1}^*}{\Delta x}\right]\right\},
\end{aligned} \tag{8–1.19}$$

where ρ^* is a provisional value of ρ and f^* is a function of ρ^*. The idea here is to use a forward-differencing method in the downwind direction as a predictor, obtain provisional values ρ^*, and then use a backward-differencing method for the value at the next timestep. This forward-backward order gives a different result from the backward-forward order, and

gives somewhat better results than the two-step Lax-Wendroff algorithm for convecting discontinuities.

In the spirit of the Richtmyer methods, the MacCormack method may be used to find provisional values at the half timestep, and then final values at the whole timestep. It can also be used over two whole timesteps, where the provisional value is taken as the value at one timestep, and the final value is taken as the value at the next timestep. For a constant-velocity, linear problem, this formula reduces to the two-step Lax-Wendroff method given in equation (8–1.15). Therefore the Lax-Wendroff calculation in Figure 4.6 shows how the MacCormack method performs on a simple convection test problem.

There are many ways to extend this method to two dimensions. For example, the forward and backward steps can be applied differently in the x and y directions. The process could also be applied cyclically over two or four complete timesteps. Besides the original references (for example, MacCormack [1971] and Kutler and Lomax [1971]), the MacCormack method is discussed in most textbooks on CFD.

8–1.4. Padé or Compact Finite-Difference Methods

Finite-difference expressions for derivatives, such as the first derivative $f' = \partial f/\partial x$, or a second derivative, $f'' = \partial^2 f/\partial x^2$, may also be represented by formulas in which the derivatives at one point are related to those at another. When this formulation is used, it leads to an algebraic system for the derivatives that cannot be written explicitly, as was done in the methods described above. Such implicit formulations have been developed for CFD and have variously been given the names *compact methods*, *spline methods*, and *operator compact implicit (OCI) methods*. All of the implicit formulas can be derived in a systematic way from a Taylor-series expansion, as described by Peyret and Taylor (1982). General references are given in Hirsch (1988).

One formulation that leads to such an implicit representation is the *rational fraction* or *Padé approximation* (Kopal 1961), in which rational functions are used to approximate the derivatives with various orders of accuracy. A variation of the Padé approximation methods, called *compact finite-difference methods*, have been described by Lele (1992). For example, we might write

$$\begin{aligned}&\beta f'_{i-2} + \alpha f'_{i-1} + f'_i + \alpha f'_{i+1} + \beta f'_{i+2} \\ &\quad = c\,\frac{f_{i+3} - f_{i-3}}{6h} + b\,\frac{f_{i+2} - f_{i-2}}{4h} + a\frac{f_{i+1} - f_{i-1}}{2h},\end{aligned} \tag{8–1.20}$$

and then the coefficients a, b, c and α, β must be related through matching coefficients in a Taylor-series expansion. Depending on how these coefficients are matched, solutions may be of different orders of accuracy. The objective is to generate difference formulas that have only high-order truncation errors and that can be used with variable grids and boundary conditions. Even though the starting point is somewhat different than that taken for the Padé and other implicit methods, the results are similar in the sense of leading to implicit formulations that require tridiagonal or pentadiagonal solutions (discussed further in Chapter 10). Further, those derivative approximations that are of the highest possible accuracy are essentially those given by the Padé approximation methods (Lele 1992).

Since the derivatives may be represented to a high order, the methods can, in principle, produce solutions of the fluid equations that are high-order accurate. Therefore these methods are subject to the same problems of producing unphysical oscillations as other classical methods. These oscillations might be damped by the addition of some type of viscosity or they may be removed by a nonlinear flux-limiting procedure as described in Sections 8–2 through 8–4.

8–1.5. Artificial Diffusion

Second-order methods, such as the Lax-Wendroff or MacCormack methods, are only slightly diffusive. They are quite dispersive, and thus they are highly susceptible to nonlinear instabilities. In Chapter 4, we saw that unphysical oscillations appear in regions of high gradients. Because of these numerical ripples, a positive definite quantity often appears to turn negative. In some problems this may be acceptable, but in most cases, including reactive flows, unphysical negative values produce unacceptable solutions.

A traditional way to deal with this problem has been to introduce extra diffusion to damp the spurious oscillations in regions of large gradients. At best, this is a compromise because numerical diffusion also spreads shocks and discontinuities over a number of computational cells to adequately damp overshoots and undershoots. Artificial-viscosity methods were first introduced by von Neumann and Richtmyer (1950), who were solving a problem with coupled continuity equations representing shock propagation. Their approach introduces an added viscosity into the momentum and energy equations and attempts to adjust this viscosity so that short wavelengths are damped while the long wavelengths are not affected much. Using these methods for reactive flows is problematic because they do not guarantee positivity for convected chemical species.

Discussions of the use of artificial viscosity to improve solutions are given in almost every textbook on CFD. Besides the original von Neumann and Richtmyer paper and general CFD references given earlier, good references to the artificial-viscosity method include Richtmyer and Morton (1967) and Potter (1973). We recommend Potter for a good introduction and Hirsch (1988, 1990) and Anderson (1995) for more advanced discussions.

8–2. Positivity, Monotonicity, and Accuracy

The numerical requirements of accuracy, causality, monotonicity, and positivity were defined and discussed in some detail in Chapter 4. In most CFD computations, accurate resolution of steep gradients by a minimal number (perhaps only a few) grid points is important: the number of grid points needed directly controls the cost of a computation, and the gradients control the dynamics of the system. Unless the computational grid resolves the flow well, flow across the grid is subject to a substantial amount of numerical diffusion which exists because of the algorithmic requirement that the solutions remain stable with no unphysical oscillations.

Monotonicity and positivity are not exactly the same, but their ramifications for convection algorithms are essentially identical. Figure 8.1 shows four profiles that could be segments of the same curve. Segment (a) is monotone because the sign of the gradient does not change, and it is positive because the sign of the function does not change. Because

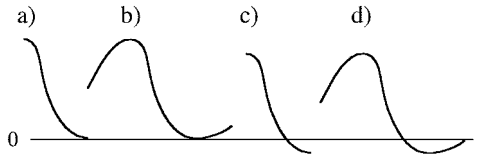

Figure 8.1. Schematic illustrating the difference between positive and monotone profiles: (a) positive and monotone, (b) positive but not monotone, (c) monotone but not positive, and (d) not monotone and not positive.

multiplying a solution of the continuity equation everywhere by minus one leaves a solution of the continuity equation, entirely negative profiles are still positive in the sense we are considering. The somewhat extended segment (b) is positive but not monotone. The segment (c) is monotone but not positive. Segment (d) is neither monotone nor positive.

The application of these terms to solution algorithms is somewhat more technical and should not be confused with the description of specific solutions. The importance of positivity arises because convection, according to equation (8–0.2) or (8–0.4), cannot take everywhere positive profiles such as those described by (a) or (b) in Figure 8.1, and produce negative values anywhere, regardless of the velocity profile. If (a) and (b) were negative profiles, convection could not make them positive. This is true in multidimensions as well as one dimension. Algorithms that reproduce this property are said to be *positive* or *positivity-preserving*.

Monotonicity is a somewhat more restrictive requirement than positivity because it concerns portions of *all* profiles, including those both near or far from extrema and those that may have negative as well as positive values. The importance of monotonicity arises because advection alone (that is, the $\mathbf{v} \cdot \nabla \rho$ term ignoring compression and source terms) cannot generate new local maxima or minima anywhere in the solution. Monotone algorithms reproduce this property. Their computed solutions do not display a large class of purely numerical ripples that are often confused with turbulence. Unfortunately, monotonicity is not really defined in multidimensions, although it can be assessed in any particular direction, such as the directions of the computational grid. This has consequences for algorithms, as discussed in Section 8–4.4. As a practical matter, the flux-limiting techniques used to ensure monotonicity generally ensure positivity, and *vice versa*, so we generally use the terms interchangeably when applied to algorithms. In principle, there can be differences.

Figure 8.2 shows how numerical diffusion enters the first-order donor-cell algorithm that was introduced in Chapter 4. A discontinuity, located at $x = 0$ at time $t = 0$, moves at a constant velocity from left to right. The velocity, v, the timestep, Δt, and computational cell size, Δx, are chosen so $v\Delta t/\Delta x = 1/3$; the discontinuity should travel one third of a cell per timestep. The solution given by the solid line in the figure uses the donor-cell algorithm.

$$\rho_i^{n+1} = \rho_i^n - \frac{v\Delta t}{\Delta x}\left(\rho_i^n - \rho_{i-1}^n\right). \tag{8–2.1}$$

If the $\{\rho_i\}$ are all positive at some time $t = n\ \Delta t$ and

$$\left|\frac{v\Delta t}{\Delta x}\right| \leq 1 \tag{8–2.2}$$

in each cell, the new density values $\{\rho_i^{n+1}\}$ at time $t = (n + 1)\Delta t$ are also positive. The

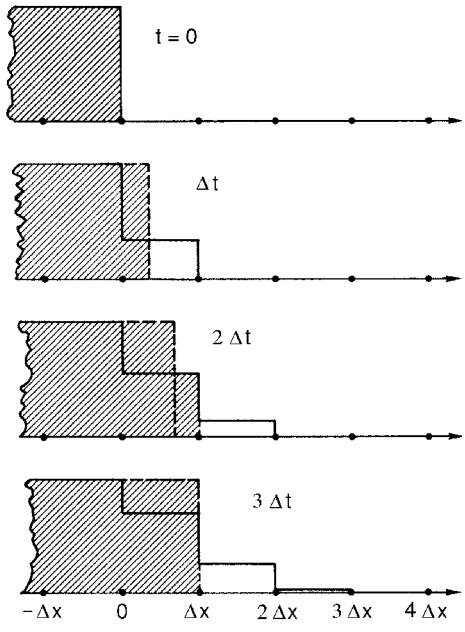

Figure 8.2. The results of convecting a discontinuity with the highly diffusive donor-cell algorithm. The heavy dashed line is the exact profile, which coincides with the numerical solution at $t = 0$. The heavy solid line is the numerical solution.

price for guaranteed positivity in this algorithm is the severe, unphysical spreading of the profile, which should be a discontinuity located at $x = vt$.

In Figure 8.2, the initial discontinuity crumbles away in a process that looks like physical diffusion, but here it arises from a purely numerical source. The numerical diffusion occurs because material that has just entered a cell, and should still be near the left boundary, is smeared over the entire cell when it is represented on the Eulerian grid. Higher-order approximations to the convective derivatives are required to reduce this diffusion. Exploiting the desired effects of numerical diffusion, often called *smoothing* or *numerical dissipation*, while circumventing the undesired effects, is a large part of the art in CFD.

Now consider a three-point, explicit finite-difference formula for advancing the set of cell or node values $\{\rho_i^n\}$ one timestep to $\{\rho_i^{n+1}\}$,

$$\rho_i^{n+1} = a_i \rho_{i-1}^n + b_i \rho_i^n + c_i \rho_{i+1}^n. \tag{8–2.3}$$

This general form includes the donor-cell algorithm and a number of other common algorithms. As before, Δx and Δt are constants. Equation (8–2.3) can be rewritten in an equivalent form that guarantees conservation,

$$\begin{aligned} \rho_i^{n+1} = \rho_i^n - \frac{1}{2}\big[&\epsilon_{i+\frac{1}{2}}(\rho_{i+1}^n + \rho_i^n) - \epsilon_{i-\frac{1}{2}}(\rho_i^n + \rho_{i-1}^n)\big] \\ &+ \big[\nu_{i+\frac{1}{2}}(\rho_{i+1}^n - \rho_i^n) - \nu_{i-\frac{1}{2}}(\rho_i^n - \rho_{i-1}^n)\big], \end{aligned} \tag{8–2.4}$$

where

$$\epsilon_{i+\frac{1}{2}} \equiv v_{i+\frac{1}{2}} \frac{\Delta t}{\Delta x}. \tag{8–2.5}$$

The values of variables at interface $i+\frac{1}{2}$ are averages of values at cells $i+1$ and i, and the values at interface $i-\frac{1}{2}$ are averages of values at cells i and $i-1$. The $\{\nu_{i+\frac{1}{2}}\}$ are nondimensional numerical diffusion coefficients which appear as a consequence of considering adjacent grid points. Conservation of ρ in equation (8–2.4) also constrains the coefficients a_i, b_i, and c_i in equation (8–2.3) by the condition

$$a_{i+1} + b_i + c_{i-1} = 1 \,. \tag{8–2.6}$$

Positivity of $\{\rho_i^{n+1}\}$ for all possible positive profiles $\{\rho_i^n\}$ requires that $\{a_i\}$, $\{b_i\}$, and $\{c_i\}$ each individually be positive for all i. As remarked earlier, monotonicity and positivity are closely related for the one-dimensional case so we will continue to use the terms rather interchangeably.

Matching corresponding terms in equations (8–2.4) and (8–2.3) gives

$$\begin{aligned} a_i &\equiv \nu_{i-\frac{1}{2}} + \frac{1}{2}\epsilon_{i-\frac{1}{2}}, \\ b_i &\equiv 1 - \frac{1}{2}\epsilon_{i+\frac{1}{2}} + \frac{1}{2}\epsilon_{i-\frac{1}{2}} - \nu_{i+\frac{1}{2}} - \nu_{i-\frac{1}{2}}, \\ c_i &\equiv \nu_{i+\frac{1}{2}} - \frac{1}{2}\epsilon_{i+\frac{1}{2}}. \end{aligned} \tag{8–2.7}$$

If the $\{\nu_{i+\frac{1}{2}}\}$ are positive and large enough, they ensure that the $\{\rho_i^{n+1}\}$ are positive. If they are too large, some of the coefficients $\{b_i\}$ can be negative. The positivity conditions derived from equations (8–2.7) are

$$\begin{aligned} &\left|\epsilon_{i+\frac{1}{2}}\right| \le \frac{1}{2}, \\ &\frac{1}{2} \;\ge\; \nu_{i+\frac{1}{2}} \ge \frac{1}{2}\left|\epsilon_{i+\frac{1}{2}}\right|, \end{aligned} \tag{8–2.8}$$

for all i. Thus the conditions in equation (8–2.8) for guaranteed *positivity* lead directly to numerical diffusion in addition to the desired convection,

$$\rho_i^{n+1} = \rho_i^n + \nu_{i+\frac{1}{2}}\left(\rho_{i+1}^n - \rho_i^n\right) - \nu_{i-\frac{1}{2}}\left(\rho_i^n - \rho_{i-1}^n\right) + \text{convection}, \tag{8–2.9}$$

where equation (8–2.8) holds. This first-order numerical diffusion rapidly smears a sharp discontinuity. When $\nu_{i+\frac{1}{2}} = \frac{1}{2}|\epsilon_{i+\frac{1}{2}}|$, the smallest diffusion still preserving positivity, the formulas (8–2.3) and (8–2.4) recover the donor-cell algorithm, which is the simplest first-order upwind method. This means that any method linear in ρ that is higher than first order cannot guarantee monotonicity for any convected profile.

If algorithms are used with $\nu_{i+\frac{1}{2}} < \frac{1}{2}|\epsilon_{i+\frac{1}{2}}|$, positivity (monotonicity) is not necessarily destroyed but can no longer be guaranteed. In practice, monotonicity is almost always violated by strong shocks and convected discontinuities unless the inequalities stated in equation (8–2.8) hold. Nevertheless, the numerical diffusion implied by equation (8–2.8) is unacceptable. The diffusion coefficient $\{\nu_{i+\frac{1}{2}}\}$, cannot be zero, because the explicit three-point formula, equation (8–2.4), is subject to the forward-differenced numerical stability problem seen with equation (8–1.8) or (8–1.10). Finite-difference methods that are higher

than first-order accurate in the mathematical sense, such as the Lax-Wendroff methods, reduce the numerical diffusion but sacrifice assured positivity.

This is a dilemma for which a remedy is needed. Introducing numerical diffusion that depends on the profile allows a way out of this problem. The resulting monotone algorithms, however, are no longer linear even though the underlying continuity equation is linear. The large diffusion fluxes required by equation (8–2.8) are only applied in narrow regions where they are needed, so that the total amount of diffusion is reduced significantly. Thus the overall methods could be second-order or higher accuracy on average, even though they are only first-order accurate in some regions.

To examine the problem of stability and positivity, we use an analysis similar to that used in Chapter 4. Consider convecting test functions of the form

$$\rho_i^n \equiv \rho_o^n e^{ii\beta}, \tag{8–2.10}$$

where

$$\beta \equiv k\ \Delta x = \frac{2\pi\ \Delta x}{\lambda}, \tag{8–2.11}$$

and i indicates $\sqrt{-1}$. Substituting this solution into equation (8–2.4) gives

$$\rho_o^{n+1} = \rho_o^n[1 - 2\nu(1 - \cos\beta) - \mathrm{i}\epsilon \sin\beta]. \tag{8–2.12}$$

Assume that the coefficients are constant, so that

$$\begin{aligned} \left\{\nu_{i+\frac{1}{2}}\right\} &= \nu \\ \left\{\epsilon_{i+\frac{1}{2}}\right\} &= \epsilon. \end{aligned} \tag{8–2.13}$$

The exact theoretical solution to this linear problem is

$$\rho_o^{n+1}\big|_{\text{exact}} = \rho_o^n e^{-\mathrm{i}kv\Delta t}. \tag{8–2.14}$$

Therefore the difference between the exact solution and equation (8–2.12) is the numerical error generated at each timestep.

The amplification factor was defined in Chapter 4 as

$$A \equiv \frac{\rho_o^{n+1}}{\rho_o^n}, \tag{8–2.15}$$

and an algorithm is always linearly stable if

$$|A|^2 \ \leq\ 1. \tag{8–2.16}$$

From equation (8–2.12),

$$|A|^2 = 1 - (4\nu - 2\epsilon^2)(1 - \cos\beta) + (4\nu^2 - \epsilon^2)(1 - \cos\beta)^2, \tag{8–2.17}$$

which must be less than unity for all permissible values of β between 0 and π. In general, $\nu > \frac{1}{2}\epsilon^2$ ensures stability of the linear convection algorithm for any Fourier harmonic of

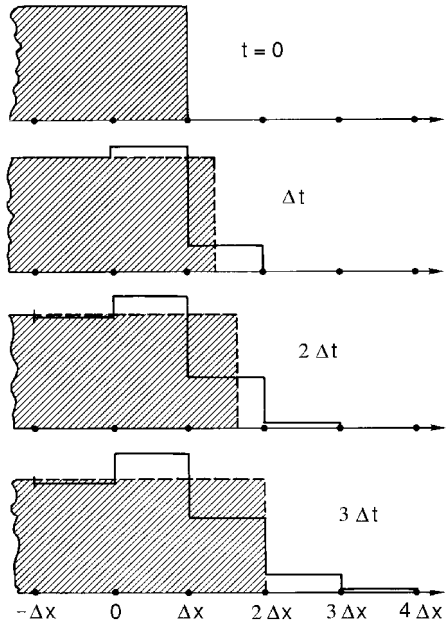

Figure 8.3. The results of convecting a discontinuity with an algorithm with enough diffusion to maintain stability, but not enough to hide the effects of dispersion. Note the growing nonphysical overshoot and the diffusive numerical precursor at times after $t = 0$.

the disturbance, provided that Δt is chosen so that $|\epsilon| \leq 1$. This stability condition is a factor of two less stringent than the positivity conditions $|\epsilon| \leq \frac{1}{2}$. When $\nu > \frac{1}{2}$, there are combinations of ϵ and β where $|A|^2 > 1$, for example, $\epsilon = 0$ with $\beta = \pi$. Thus the range of acceptable diffusion coefficients is quite closely prescribed,

$$\frac{1}{2} \geq \nu \geq \frac{1}{2}|\epsilon| \geq \frac{1}{2}\epsilon^2. \tag{8–2.18}$$

Even the minimal numerical diffusion required for linear stability, $\nu = \frac{1}{2}\epsilon^2$, may be substantial when compared to the physically correct diffusion effects such as thermal conduction, molecular diffusion, or viscosity. Figure 8.3 shows the first few timesteps from the same test problem as in Figure 8.2, but using $\nu = \frac{1}{2}\epsilon^2$ rather than $\nu = \frac{1}{2}\epsilon$ required for positivity. The spreading of the profile is only one third as much as in the previous case where positivity was assured linearly, but a numerical precursor still reaches two cells beyond the correct discontinuity location. Furthermore, the overshoot between $x = -\Delta x$ and $x = 0$ in Figure 8.3 is a consequence of underdamping the solution. The loss of *monotonicity* indicated by the overshoot can be as bad as violating positivity. A new, unphysical maximum in ρ has been introduced into the solution. When the convection algorithm is stable, but not positive, the numerical diffusion is not large enough to mask either numerical dispersion or the Gibbs phenomenon, so the solution is no longer necessarily monotone. New ripples, that is, new maxima or minima, are introduced numerically.

From the discussion above, it would appear that the requirements of positivity, monotonicity, and accuracy are mutually exclusive. *Nonlinear monotone methods*, as suggested above, were invented to circumvent this dilemma. These methods use the marginally stabilizing $\nu = \frac{1}{2}\epsilon^2$ diffusion where monotonicity is not threatened, and increase ν to values

approaching $\nu = \frac{1}{2}|\epsilon|$ when required to assure that the solution remains monotone. Different criteria are imposed in the same timestep at different locations on the computational grid according to the local profiles of the physical solution. *The dependence of the local smoothing coefficients ν on the solution profile makes the overall algorithm nonlinear and allows the net amount of numerical diffusion to be far below the $\nu = \frac{1}{2}\epsilon^2$ level required to guarantee monotonicity in a linear algorithm.*

To prevent numerically induced, nonmonotone behavior in ρ, which could arise from dispersion or Gibbs errors, a minimum amount of numerical diffusion must be added to assure positivity and stability at each timestep. We write this minimal diffusion as

$$\nu \approx \frac{|\epsilon|}{2}\,(c + |\epsilon|) \tag{8–2.19}$$

where c is a *clipping factor*, $0 \leq c \leq 1-|\epsilon|$, controlling how much extra diffusion, over the stability limit, must be added to ensure positivity. In the vicinity of steep discontinuities, $c \approx 1 - |\epsilon|$, and in smooth regions away from local maxima and minima, $c \approx 0$. Remember that much of this diffusion is counteracting the destabilizing time integration, so the net diffusion in the algorithm is really much less than implied by equation (8–2.18).

Nonlinear monotone algorithms are a reliable, robust way to calculate convection. The flux-limiting process ensures that the numerical smoothing is directly coupled to the evolving flow profile as well as the truncation errors in the underlying linear convection algorithm and thus enforces physically and mathematically meaningful constraints on the solutions. Short wavelength structure, that cannot be well resolved numerically in the evolving solutions, is strongly damped. In Chapter 12, we show how this dissipation remaining in algorithms after the flux-limiting process could constitute a minimal subgrid turbulence model when these algorithms are used to solve the coupled systems of fluid dynamics equations.

The first specifically monotone positivity-preserving technique was the flux-corrected transport algorithm developed by Boris and Book (Boris, 1971; Boris and Book, 1973, 1976a,b). This was followed closely by the work of Van Leer (1973, 1979) and Harten (1974, 1983). There have been a series of other monotone methods, several of which are described in Section 8.3. General references to these methods are given in Colella and Woodward (1984), Woodward and Colella (1984), Rood (1987) and LeVeque (1992). In addition to the monotonicity-preserving flux limiter, some of the recent nonlinear monotone methods incorporate problem-specific analytic approximations to the physical waves encountered in the flow. For this reason they are treated in greater depth in the next chapter.

8–3. Monotone Convection Algorithms and Flux Limiting

This section and Section 8–4 show how the constraints of conservation, causality, and positivity may be imposed to develop monotone algorithms. Flux-corrected transport is a simple monotone algorithm and will be used as an example to guide discussion. FCT is a general approach to solving continuity equations and led to a number of nonlinear monotone, positive, or *TVD* algorithms. The good properties of FCT and other monotone

algorithms are achieved partly through the choice of coefficients in the representation of the fluxes and partly through a process called *flux correction* or *flux limiting*. For a given grid resolution, a good monotone algorithm can be about an order of magnitude more accurate than the simple first- and second-order linear finite-difference methods described earlier. Though originally developed for time-dependent shock and blast computations, monotone algorithms have now been widely adopted for aerodynamics and steady-state Navier-Stokes computations.

8–3.1. The Basic Idea of Flux-Corrected Transport

Now rewrite the explicit three-point approximation to the continuity equation given in equation (8–2.3) to determine provisional values, $\tilde{\rho}$, in terms of the previous timestep or "old" values, ρ^o,

$$\tilde{\rho}_i = a_i \rho^o_{i-1} + b_i \rho^o_i + c_i \rho^o_{i+1}. \tag{8–3.1}$$

Again, equation (8–2.6) must be satisfied for conservation and $\{a_i\}$, $\{b_i\}$, and $\{c_i\}$ must all be greater than or equal to zero to assure positivity.

Equation (8–3.1) in conservative form, as with equation (8–2.4), becomes

$$\begin{aligned} \tilde{\rho}_i &= \rho^o_i - \frac{1}{2}\left[\epsilon_{i+\frac{1}{2}}\left(\rho^o_{i+1} + \rho^o_i\right) - \epsilon_{i-\frac{1}{2}}\left(\rho^o_i + \rho^o_{i-1}\right)\right] \\ &\quad + \left[\nu_{i+\frac{1}{2}}\left(\rho^o_{i+1} - \rho^o_i\right) - \nu_{i-\frac{1}{2}}\left(\rho^o_i - \rho^o_{i-1}\right)\right] \\ &= \rho^o_i - \left[f_{i-\frac{1}{2}} - f_{i+\frac{1}{2}}\right]. \end{aligned} \tag{8–3.2}$$

In every cell i, $\tilde{\rho}_i$ differs from ρ^o_i as a result of the inflow and outflow of the fluxes of ρ, denoted $f_{i\pm\frac{1}{2}}$. The fluxes, here defined to have the same units as ρ, are successively added and subtracted along the array of densities ρ^o_i. In this way, overall conservation of ρ is satisfied by construction. Summing all the provisional densities gives the sum of the old densities. The expressions in brackets involving $\epsilon_{i\pm\frac{1}{2}}$ are called the *convective fluxes*.

By comparing equations (8–3.2) and (8–3.1), we obtain the conditions relating the a, b, and c's to the ϵ's and ν's, as in equation (8–2.7). In equation (8–3.2), the $\{\nu_{i+\frac{1}{2}}\}$ are dimensionless numerical diffusion coefficients included to ensure positivity of the provisional values $\{\tilde{\rho}_i\}$. The positivity condition for the provisional $\{\tilde{\rho}_i\}$ is given in equation (8–2.8).

The two sets of coefficients remaining in equation (8–3.2) are still to be determined. One of these must ensure an accurate representation of the mass flux terms. Thus

$$\epsilon_{i+\frac{1}{2}} = v_{i+\frac{1}{2}} \frac{\Delta t}{\Delta x}, \tag{8–3.3}$$

where $\{v_{i+\frac{1}{2}}\}$ is the fluid velocity approximated at the cell interfaces. The other set of coefficients, $\{\nu_{i+\frac{1}{2}}\}$, can be chosen to maintain positivity and stability.

The provisional values $\tilde{\rho}_i$ must be strongly diffused to ensure positivity. If $\nu_{i+\frac{1}{2}} = \frac{1}{2}|\epsilon_{i+\frac{1}{2}}|$ in equation (8–2.8), we have the diffusive donor-cell algorithm. A correction in the algorithm to remove most of this strong diffusion uses an additional *antidiffusion* stage,

$$\rho^n_i = \tilde{\rho}_i - \mu_{i+\frac{1}{2}}(\tilde{\rho}_{i+1} - \tilde{\rho}_i) + \mu_{i-\frac{1}{2}}(\tilde{\rho}_i - \tilde{\rho}_{i-1}), \tag{8–3.4}$$

to get the new values of $\{\rho_i^n\}$. Here $\{\mu_{i+\frac{1}{2}}\}$ are positive *antidiffusion coefficients*. Antidiffusion reduces the strong diffusion implied by equation (8–2.8) but also reintroduces the possibility of negative values or unphysical overshoots in the new profile. Further, if the values of $\{\mu_{i+\frac{1}{2}}\}$ are too large, the new solution $\{\rho_i^n\}$ will be unstable.

To obtain a monotonicity-preserving algorithm, modify the antidiffusive fluxes in equation (8–3.4) by a process called *flux correction* or *flux limiting*. The antidiffusive fluxes,

$$f^{ad}_{i+\frac{1}{2}} \equiv \mu_{i+\frac{1}{2}}\,(\tilde{\rho}_{i+1} - \tilde{\rho}_i), \tag{8–3.5}$$

appearing in equation (8–3.4) are *limited* as described below to ensure positivity and stability.

The largest antidiffusion coefficients $\{\mu_{i+\frac{1}{2}}\}$ that still guarantee positivity are

$$\mu_{i+\frac{1}{2}} \approx \nu_{i+\frac{1}{2}} - \frac{1}{2}\,\left|\epsilon_{i+\frac{1}{2}}\right|. \tag{8–3.6}$$

This is still too diffusive. To reduce the residual diffusion $(\nu - \mu)$ further, the flux correction must be nonlinear and depend on the actual values and shape of the density profile $\{\tilde{\rho}_i\}$.

The idea behind nonlinear flux-correction formulas is as follows. Suppose the density $\tilde{\rho}_i$ at grid point i reaches zero while its neighbors are positive. Then the second derivative is locally positive and any antidiffusion would force the minimum density value $\tilde{\rho}_i = 0$ to be negative. Because this cannot be allowed physically, the antidiffusive fluxes should be limited so minima in the profile are made no deeper by the antidiffusive stage of equation (8–3.4). Because the continuity equation is linear, we could equally well solve for $\{-\rho_i^n\}$. Hence, we also must require that antidiffusion not make the maxima in the profile any larger. These two conditions form the basis for FCT and also are at the core of other monotone methods. *"The antidiffusion stage should generate no new maxima or minima in the solution, nor should it accentuate already existing extrema"* (Boris and Book 1973).

This qualitative idea for nonlinear filtering can be quantified. The new values $\{\rho_i^n\}$ are given by

$$\rho_i^n = \tilde{\rho}_i - f^c_{i+\frac{1}{2}} + f^c_{i-\frac{1}{2}}, \tag{8–3.7}$$

where the corrected fluxes $\{f^c_{i+\frac{1}{2}}\}$ satisfy

$$f^c_{i+\frac{1}{2}} \equiv S \cdot \max\left\{0,\ \min\left[S \cdot (\tilde{\rho}_{i+2} - \tilde{\rho}_{i+1}),\ \left|f^{ad}_{i+\frac{1}{2}}\right|,\ S \cdot (\tilde{\rho}_i - \tilde{\rho}_{i-1})\right]\right\}. \tag{8–3.8}$$

Here $|S| = 1$ and sign $S \equiv$ sign $(\tilde{\rho}_{i+1} - \tilde{\rho}_i)$.

To see what this flux-correction formula does, assume that $(\tilde{\rho}_{i+1} - \tilde{\rho}_i)$ is greater than zero. Then equation (8–3.8) gives either

$$\begin{aligned} &f^c_{i+\frac{1}{2}} = \min[(\tilde{\rho}_{i+2} - \tilde{\rho}_{i+1}),\ \mu_{i+\frac{1}{2}}(\tilde{\rho}_{i+1} - \tilde{\rho}_i),\ (\tilde{\rho}_i - \tilde{\rho}_{i-1})] \quad \text{or} \\ &f^c_{i+\frac{1}{2}} = 0, \end{aligned} \tag{8–3.9}$$

whichever is larger. The "raw" antidiffusive flux, $f^{ad}_{i+\frac{1}{2}}$ given in equation (8–3.5), always tends to decrease ρ^n_i and to increase ρ^n_{i+1}. The flux-limiting formula ensures that the corrected flux cannot push ρ^n_i below ρ^n_{i-1}, which would produce a new minimum, or push ρ^n_{i+1} above ρ^n_{i+2}, which would produce a new maximum. Equation (8–3.8) is constructed to take care of all cases of sign and slope.

The formulation of a simple, one-dimensional FCT algorithm therefore consists of the following four steps:

1. Compute the transported and diffused values $\tilde{\rho}_i$ from equation (8–3.2), where $\nu_{i+\frac{1}{2}} > \frac{1}{2}|\epsilon_{i+\frac{1}{2}}|$ satisfies the monotonicity condition. Add in any additional source terms, for example, $-\nabla P$.
2. Compute the raw antidiffusive fluxes from equation (8–3.5).
3. Correct (limit) these fluxes using equation (8–3.8) to assure monotonicity.
4. Perform the indicated antidiffusive correction through equation (8–3.7).

Steps 3 and 4 are the new components introduced by FCT. There are many modifications of this prescription that accentuate various properties of the solution. Some of these are summarized in Boris and Book (1976a,b) and Zalesak (1979, 1981), and more recently by Boris et al. (1993).

8–3.2. Generalizations and Extensions of FCT Algorithms

Many variations and generalizations of FCT have been developed since its introduction in 1971. The essential concept of monotonicity and positivity introduced by FCT is an integral part of subsequent algorithms, including MUSCL, TVD, and PPM. Besides their somewhat different formulations of flux limiting, many of these algorithms have additional constraints on the physics or mathematical form. These are discussed further in Chapter 9 and in more detail in the references given.

FCT algorithms with spectral accuracy in the underlying convection algorithm have been published by Zalesak (1981) and McDonald, Ambrosiano, and Zalesak (1985). The nonlinear flux-correction formula removes the large, diffusive, amplitude errors needed to ensure stability and positivity. The next largest source of errors are phase errors that arise because the templates of the convective derivatives are local. Spectral algorithms are the limiting cases with "infinite" order of accuracy, and they provide a mechanism for evaluating the importance of phase errors. Boris and Book (1976b) developed a Fourier FCT algorithm to minimize both phase and amplitude errors. To make the algorithm reversible, it was necessary to add implicit diffusion to the underlying algorithm, so that the antidiffusion could be performed locally. The most accurate solutions of the standard square-wave test problem, with linear stability for every harmonic, were obtained with this version of FCT.

A finite-element version of FCT was developed by Löhner and Patnaik (1987) and Löhner et al. (1987a,b). The extension of FCT to generalized meshes opened the way to simulations with complex, time-evolving geometries using monotone algorithms. It was also recognized by Karniadakis (1998) that complex geometry could be accommodated with higher-order algorithms than finite-element algorithms afforded by adding a

monotone flux limiter to spectral-element methods. Development of the virtual cell embedding algorithms (Landsberg et al. 1993; Landsberg and Boris 1997) extended the complex geometry capabilities of FCT to the simplicity and high accuracy of regular Cartesian meshes.

A notable modification of FCT was given by Odstrčil (1990), who showed how the half-step/whole-step integration procedure, used in the solution of coupled continuity equations representing fluid dynamics, could be modified to allow a factor of two increase in the computational timestep. This modification comes at the cost of some additional computer memory, but it can be valuable for explicit fluid dynamics problems where the presence of other chemical or physical processes do not require further reductions of the global timestep. Granjouen (1990) derived a modification of the FCT formulation in which the diffusion and antidiffusion coefficients depend on velocity gradients. The solutions have reduced diffusion and noise properties at the cost of additional computer memory and some extra computing.

As with other CFD algorithms, it is sometimes advantageous to build additional physics directly into FCT algorithms. When the main variability of the solution occurs as deviations from a known state, rewriting the convected variables can sometimes give improved results. For example, a modified form of flux correction can be used for atmospheric problems to describe variables that decrease roughly exponentially with height. The prescription for no new minima or maxima permits ripples or "terraces" to form on the side of steep gradients. This effect is largely eliminated when the underlying exponential variation is divided out of the variables before differences are computed for the flux limiter. Conversely, this same effect also means that explicit solution of the cylindrical or spherical one-dimensional radial equations is highly preferable to redefining the fundamental variables to include the radial factors. Even though defining $\rho'(\mathbf{r}, \mathbf{t}) \equiv \mathbf{r}\rho(\mathbf{r}, \mathbf{t})$ allows a cylindrical problem to be treated as if it were Cartesian, it also allows unphysical new maxima in $\rho(\mathbf{r}, \mathbf{t})$ that are masked from the flux correction formula by the radial factor.

In Chapter 9, we discuss an implicit version of FCT, called the *barely implicit correction*, or BIC, for use in reactive flows when the fluid velocity is much smaller than the sound speed (Patnaik et al. 1987). By solving an elliptic equation for the pressure field, the limit imposed by the sound speed on the computational timestep is removed. Then, for certain types of fluid problems in which acoustic effects are not important, the fluid dynamics timestep could be two or three orders of magnitude larger. Patnaik points out that the BIC process is general, and it can be combined with any continuity-equation solver, not only FCT. BIC is an important computational capability for several reasons. The first is that BIC, combined with a monotone method such as FCT, produces an implicit solver that maintains the advantages of using a nonlinear monotone convection algorithm. The second is that it solves a compressible flow: it can include the effects of a finite $\nabla \cdot \mathbf{v}$ term that arises from chemical heat release. The third is that it has been parallelized successfully for multidimensional flows (Moon et al. 1995). Thus it is particularly useful for algorithms used for reactive flows.

Weber, Boris, and Gardner (1979) generalized FCT to solve the magnetohydrodynamic (MHD) equations. Later, DeVore (1989) developed a fully two-dimensional FCT module that was later generalized to three dimensions. Now DeVore and colleagues have generalized the FCT treatment of continuity equations to include MHD problems in which

it is important to maintain the divergence constraint $\nabla \cdot \mathbf{B} = 0$ (Devore 1991; Karpen, Antiochos, and DeVore 1996; Parhi et al. 1996; Murawski et al. 1996; Antiochos, DeVore, and Klimchuk 1999).

Another area of development is that of algorithms for low-velocity MHD systems. Explicit MHD integration algorithms are subject to a numerical timestep limitation determined by the magnetoacoustic speed. In many reactive MHD phenomena, however, the important physical timescales are controlled by slower flows. (One application of this would be computing the movement of magnetic footpoints by photospheric convection on the sun.) To relax this timestep constraint, DeVore (1991) has developed an implicit treatment of only the Alfvenic transport terms. This extends BIC to MHD applications.

FCT methods have causality built in. This is accomplished by evaluating the derivatives locally and using a conservative, telescoping formulation such as equation (8–2.4). Causality means that all fluxes between two cells A to B must pass through the cells in between. Equivalent physical ideas are central to relativity theory so we expect that relativistic formulations of FCT and other monotone algorithms are possible. Further, these can be important since numerical methods spread information across a single cell essentially instantaneously and thus violate physical causality and speed limitations on a grid scale. A special relativistic, two-dimensional MHD model was developed based on FCT by Dubal (1991). MaMarti and Müller (1996) extended this to a PPM algorithm, although not for MHD, giving a simple prescription that should be applicable to a number of fluid dynamics algorithms.

An important contribution to both understanding and application was introduced by Zalesak (1979), who showed how FCT can be implemented as a combination of a monotone low-order algorithm with a nonmonotone high-order algorithm. This approach, which has features similar to the hybridization scheme of Harten and Zwas (1972), generalized the algorithm by giving a prescription for how nonmonotone methods may be made monotone. Another important result of his approach is in showing how FCT could be made fully multidimensional, thereby avoiding some of the problems that arise when direction splitting is used. The later work by DeVore (1998) delved more deeply into the issues of differencing and flux limiting for fully multidimensional computations, some of which are discussed below. The result is a new flux-limiting prescription for multidimensional FCT models that improves the performance of the algorithms for both fluid dynamic and MHD systems.

8–3.3. The Effects of Order on FCT

In Chapter 4 we used several examples of calculations of the convection of a square wave to test various simple algorithms for flows with constant velocity. Now we return to this problem, adding two more linear tests: a sharp Gaussian profile and a half dome. These tests are important for several reasons. The Gaussian shows the effects of *clipping* the solution, and the half dome shows the effects of *terracing*. The test problems are:

1. The square wave, initially twenty cells wide, is convected at a Courant number of 0.2, that is,

$$\frac{v\,\Delta t}{\Delta x} = 0.2,$$

and we look at the profile after 800 steps. The maximum value of ρ is 2.0 and the minimum value is 0.5.

2. The Gaussian has half width $2\Delta x$. We convect this profile at a Courant number of 0.1 and examine the profile after 600 steps. The maximum value of ρ is 1.6 and the minimum is 0.1.
3. The half dome is a total of thirty cells wide. We convect the profile at a Courant number of 0.1 and examine the profile after 600 steps. The maximum value of ρ is 1.6 and the minimum is 0.1.

These three different profiles are often used to demonstrate the properties of different algorithms for solving continuity equations (Hirsch 1990; Huynh 1995; Suresh and Huynh 1997). Figure 8.4 shows the results of using second-order, fourth-order, and sixteenth-order FCT algorithms to convect these profiles. The exact solutions are the light, solid line, and the computed solutions are the dots. Such variable-order FCT methods are described by Zalesak (1981). The fourth-order method is essentially the same as the default algorithm in standard LCPFCT described in Section 8–4.5.

Figure 8.4 may be discussed in terms of effects of the phase and dispersive errors and the specific properties enforced by the flux-limiting procedure. These errors and limits interact and are manifest here as *clipping* and *terracing*. *Clipping* is a result of the nonlinear flux limiting. It is an expression of the fundamental uncertainty that arises

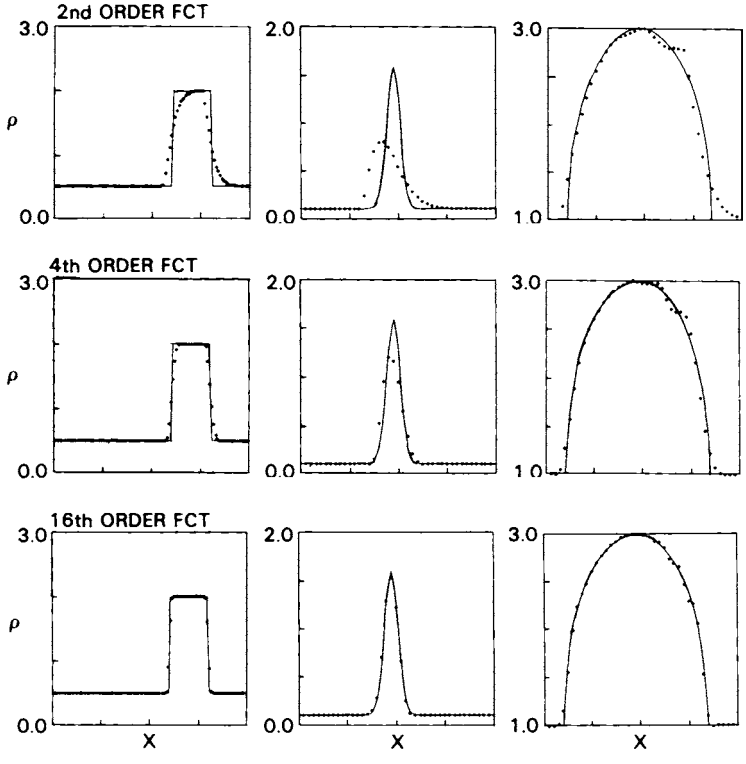

Figure 8.4. Tests of linear convection for varying order of phase accuracy using an FCT algorithm (courtesy of S. Zalesak).

due to having a finite computational cell size. Because we cannot know what is happening between grid points, and the algorithm does not allow the creation of new extrema, a local extremum cannot be propagated across a grid without loss of amplitude. Removing this property of clipping would defeat the good effects of flux limiting. *Terracing* is an integrated, nonlinear effect of residual phase errors. Figure 4.6 shows the ripples that appear in the Lax-Wendroff solution because the short wavelengths move more slowly than they should. Terracing is effectively how flux-limiting clips ripples. Neither terracing nor clipping are peculiar to FCT, but occur in other nonlinear monotone methods, such as PPM, MUSCL, or TVD. The extent to which they are a problem usually depends on the order of the solution and the particular properties of the flux limiter.

The first observation drawn from Figure 8.4 is that, for the square-wave test problem, the solutions are all large improvements over the nonmonotone solutions described in Chapter 4. The next point is that for a monotone method, specifically for FCT shown here, the order of the method, at least up to fourth order, has a strong affect on the solution. The second-order algorithm suffers a dispersive phase lag that slows the short wavelengths, severely disrupting the shape of narrow convected profiles. This is not a serious problem in the fourth-order algorithm, for which clipping errors at the steep gradients begin to exceed the phase errors.

The Gaussian test problem has a single moving maximum and so is a better demonstration and test of clipping associated with the flux limiter. The maximum is clipped less as the order of the algorithm increases. The higher-order methods also do progressively better in decreasing phase errors. The key to good performance here is the application of some form of nonlinear flux limiting to an underlying high-order algorithm, so that phase errors are not the largest errors in the solution.

When there is a long, gradual gradient, such as on the slope of the half dome, methods that remove most of the first-order diffusion show terracing effects. In Figure 8.4, terracing is noticeable mainly on the right shoulder of the dome. The solution tends to set up terraces because the slope of the profile is nonmonotone instead of smoothly varying. Selectively adding diffusion could reduce terracing, but the solution would become more diffusive.

The key to obtaining accurate solutions is maintaining conservation, positivity (monotonicity), and using a flux-correcting limiter to minimize numerical diffusion. From our experience, there is little advantage to increasing the accuracy of the phases above fourth order, although improvements in going from second to fourth order are significant. The computational cost goes up with order and there is proportionately less gained. Formal order is advantageous in these linear convective problems, but it is not clear that it has much meaning for nonlinear multidimensional problems. Further, it is also not clear how to evaluate order with variably spaced and changing grids.

The Fourier FCT algorithm (Boris and Book 1976b) was designed to have spectral accuracy (no phase errors) and be time reversible (before the flux limiter is applied). Therefore, there were neither phase errors nor amplitude errors in the underlying linear algorithm, despite the adjustable diffusion and antidiffusion allowed. The absolute error measure used in those square-wave tests was the lowest obtained for any stable algorithm. Nonetheless, the finite-resolution Gibbs errors were still present, so perfect convection of the profiles is not possible.

Ensuring monotonicity results in some rounding at the shoulder of the convected square wave. Thus a continuous, high-order interpolation of the solution between the grid point values has no additional extrema beyond those evident initially on the grid. The Fourier FCT algorithm, while not efficient or general enough for practical use, allows a close approximation to the optimal solution for convection of the square wave. A small amount of instability in the underlying linear algorithm, often called *steepening*, produces solutions with steeper gradients and sharper contact surfaces (Cheatham, Oran, and Boris n.d.). The combination of the nonlinear monotone flux-limiter and the conservation constraints prevents the steepened algorithms from displaying the worst manifestations of the instability. As with all trade-offs, however, such algorithms have other problems arising from the lurking instability, so be careful when you use them.

8–3.4. Other Approaches to Monotonicity

The following statement was first shown to be true by Godunov (1959): *There are no linear second-order or higher-order methods that guarantee monotonicity*. First-order methods, such as the Lax or donor-cell methods, guarantee positivity (they are monotone) but are extremely diffusive. Second-order linear methods, such as the Lax-Wendroff or MacCormack methods, are less diffusive, and they are susceptible to nonlinear instabilities and cause unphysical oscillations in regions of high gradients. Some techniques to enforce conservation can also cause a positive definite quantity to become negative.

Introducing diffusion into the algorithm is one traditional way to damp numerically generated oscillations in regions of large gradients. This is a compromise approach. Numerical dissipation is added into the method so that discontinuities and sharp gradients are spread over distances that can be resolved on a practical computational mesh. If the oscillations are totally damped, the solution shows the same massive diffusion as the linear first-order methods. On the other hand, if the oscillations are not totally damped, little has been gained. This is the fundamental dilemma of linear methods.

A high-resolution monotone method is at least second-order, yet still gives well-resolved nonoscillatory solutions in the vicinity of discontinuities and sharp gradients. FCT tries to achieve this by limiting fluxes and imposing physical constraints to avoid unphysical oscillations. There are a number of other approaches that try to do this, and they may be categorized in several ways. Below we list some of the categories and try to explain the essence of what they are doing. This is not meant to be inclusive, but to help explain concepts and the terminology that has become standard.

Hybridization Methods

In hybridization methods, two or more methods are blended to take advantage of the strong points of each. The term *linear hybridization* was originally used by Harten (1974) to mean the flux-by-flux weighted average of high-order and low-order fluxes. Though FCT was not originally cast in this form, Zalesak (1979) noted that FCT can be described in these terms, showing that the switches used to perform the hybridization between two linear methods reduce to nonlinear formulas of the FCT flux-limiting type. The low-order method in FCT hybridization is the same as the high-order method with added diffusion. This may be important because using a different, truly lower-order method can

introduce additional, unintended phase errors in the process of correcting local excursions from monotonicity. There are a number of other numerical methods based on combining high-order and low-order methods or fluxes, including the FCT, TVD (discussed later in this section), and the Godunov methods (Chapter 9). The term "hybridization method" may be of historical interest, but it is now misleading and adds to the general confusion.

Upwind Schemes or Upwinding

The objective of upwinding is to ensure the correct propagation of information according to that specified by the characteristics in the flow. For simplicity, the continuity equation can be rewritten as

$$\frac{\partial \rho}{\partial t} + \frac{\partial f}{\partial \rho}\frac{\partial \rho}{\partial x} = \frac{\partial \rho}{\partial t} + J\frac{\partial \rho}{\partial x} = 0, \tag{8–3.10}$$

where $J \equiv \partial f/\partial \rho$ is the Jacobian. The eigenvalues of J represent the velocity and direction of propagation of information. There are a number of ways this information can be used to improve numerical convection algorithms. *Upwinding* uses a form of the spatial derivative that is biased in the direction from which the information is coming. The idea of upwinding was introduced in the early papers by Courant, Isaacson, and Reeves (1952) and Godunov (1959), and was used in the donor-cell methods (see Roache [1982] for an historical analysis). Van Leer (1986) introduced the concepts into modern higher-order algorithms. An introduction to upwind methods is given by Anderson (1995) and more specific discussions are given by Fletcher (1988) and Hirsch (1988, 1990). The generalization of this is *flux-vector splitting*, which is based on the premise that there is more information than the velocity that can be used to improve the solution: we can take advantage of other characteristics of the flow. We return to these topics in Chapter 9.

Total-Variation-Diminishing Methods

TVD methods represent an effort to place the study of monotone, flux-limiting methods for solving hyperbolic systems of equations on a rigorous mathematical foundation. TVD is based on the observation that, given a continuity equation of the form of equation (8–0.2) or (8–0.3), and assuming that both ρ and the derivative $\partial\rho/\partial x$ are known at a given time, there is a property of the physical solutions that $|\partial\rho/\partial x|$ integrated over the entire x domain does not increase with time. The integrated quantity is called the *total variation* (*TV*), and is written as

$$TV = \int \left|\frac{\partial \rho}{\partial x}\right| dx. \tag{8–3.11}$$

TV does not increase with time for any physically correct solution. In terms of the numerical solution, the derivative can be discretized as, for example, $(\rho_{i+1} - \rho_{i-1})/2\Delta x$, then the total variation in x can be written as x

$$TV(\rho) \equiv \sum_i \frac{|\rho_{i+1} - \rho_{i-1}|}{2}. \tag{8–3.12}$$

If $TV(\rho^{n+1})$ and $TV(\rho^n)$ are evaluated at time $n+1$ and n, respectively, the algorithm is total-variation diminishing if

$$TV(\rho^{n+1}) \leq TV(\rho^n). \tag{8–3.13}$$

Achieving this requires enforcing some type of flux limiting. As in FCT and other nonlinear monotone methods, TVD schemes introduce nonlinear functions that force the TVD condition. These functions are intended to limit the gradients by modifying the flux terms in the difference equations. Thus TVD methods all incorporate forms of flux limiting. To paraphrase Harten, TVD is a monotone method, and a monotone method is TVD. Zalesak (1987) has done extensive work to examine the basic ideas and extract the flux-limiting procedures in many of these methods. Original contributors to the development of TVD include Harten (1983), Davis (1984), Yee, Warming, and Harten (1983), Sweby (1984), and Chakravarthy and Osher (1985a,b). The philosophy, use, and generalization of TVD methods are discussed in many texts on CFD (see, for example, Hirsch [1990] and Tannehill et al. [1997]).

Slope Limiting and Flux Limiting

A geometrical approach to flux limiting is called *slope limiting*. The idea here is to replace, for example, a constant piecewise representation by some more accurate representation, such as piecewise linear. Even greater accuracy may be obtained by using a quadratic form, as is done in the piecewise-parabolic method (Colella and Woodward 1984) or the higher-order reconstructions used in the essentially nonoscillatory methods (Harten and Osher 1987; Harten et al. 1987). In fact, any flux limiter can be converted into a slope limiter in the same way that any slope limiter can be converted into a flux limiter. The topic of slope limiting is discussed by LeVeque (1992), who presents an excellent discussion of the relationship between these approaches.

8–4. Coupled Sets of Continuity Equations

There are several approaches to solving multidimensional continuity equations. Above we discussed extending the one-dimensional FCT algorithm to two and three dimensions. Perhaps the easiest way to do this, both in terms of ease of understanding and implementation, is based on direction splitting. This is similar to the process-splitting approach to be described in Chapter 11. The integrations for the different spatial directions are done separately, and the results then coupled. Below we summarize the principles of direction (or timestep) splitting and then proceed to extend the discussion to coupled sets of continuity equations for density, momentum, and energy to solve the equations of fluid dynamics.

8–4.1. Lax-Wendroff Methods

In Chapter 4, a simple Lax-Wendroff method was described to solve the moving square-wave problem. Earlier in this chapter, this method was extended to solve a single general continuity equation in one dimension. Now we show how to solve systems of coupled continuity equations (Lax and Wendroff 1960, 1964) that may be used to model compressible flows with shocks.

The fluid dynamic equations in one dimension may be written in vector form as

$$\frac{\partial \mathbf{u}}{\partial t} + \frac{\partial}{\partial x}\mathbf{f}(\mathbf{u}) = 0, \tag{8–4.1}$$

where $\mathbf{u}$ and $\mathbf{f}$ are defined as vectors

$$\mathbf{u} \equiv \begin{pmatrix} \rho \\ \rho v \\ E \end{pmatrix} \quad \text{and} \quad \mathbf{f} \equiv \begin{pmatrix} \rho v \\ P + \rho v^2 \\ v(E+P) \end{pmatrix}. \tag{8–4.2}$$

The Lax-Wendroff method uses a Taylor-series in time, as in equation (8–1.2),

$$\mathbf{u}_i^{n+1} = \mathbf{u}_i^n + \Delta t \left.\frac{\partial \mathbf{u}}{\partial t}\right|_i^n + \frac{\Delta t^2}{2} \left.\frac{\partial^2 \mathbf{u}}{\partial t^2}\right|_i^n + \cdots, \tag{8–4.3}$$

and the expansion

$$\frac{\partial^2 \mathbf{u}}{\partial t^2} = -\frac{\partial}{\partial t}\frac{\partial \mathbf{f}}{\partial x} = -\frac{\partial}{\partial x}\frac{\partial \mathbf{f}}{\partial t} = -\frac{\partial}{\partial x}\left(\mathbf{A}\cdot\frac{\partial \mathbf{u}}{\partial t}\right) = \frac{\partial}{\partial x}\left(\mathbf{A}\cdot\frac{\partial \mathbf{f}}{\partial x}\right), \tag{8–4.4}$$

where $\mathbf{A}(\mathbf{u})$ is the Jacobian of $\mathbf{f}(\mathbf{u})$ with respect to $\mathbf{u}$, that is

$$A_{kj} \equiv \frac{\partial f_k}{\partial u_j}. \tag{8–4.5}$$

Writing the partial derivatives as finite differences, we solve for $\mathbf{u}_i^{n+1}$,

$$\begin{aligned} \mathbf{u}_i^{n+1} = \mathbf{u}_i^n &- \frac{1}{2}\frac{\Delta t}{\Delta x}\left(\mathbf{f}_{i+1}^n - \mathbf{f}_{i-1}^n\right) \\ &+ \frac{1}{2}\left(\frac{\Delta t}{\Delta x}\right)^2 \left[\mathbf{A}_{i+\frac{1}{2}}^n \cdot \left(\mathbf{f}_{i+1}^n - \mathbf{f}_i^n\right) - \mathbf{A}_{i-\frac{1}{2}}^n \cdot \left(\mathbf{f}_i^n - \mathbf{f}_{i-1}^n\right)\right] \end{aligned} \tag{8–4.6}$$

where

$$\mathbf{A}_{i+\frac{1}{2}}^n = \mathbf{A}\left(\frac{1}{2}\mathbf{u}_{i+1}^n + \frac{1}{2}\mathbf{u}_i^n\right). \tag{8–4.7}$$

Equations (8–4.6) and (8–4.7) generalize equations (8–1.8) through (8–1.11) to a system of coupled continuity equations in one dimension.

Richtmyer (1963) has shown that these methods can be written in two-step forms in which the first step is a first-order Lax method and the second step is a centered leapfrog method (see Section 8–1.2). The generalization of equation (8–1.15) is

$$\begin{aligned} \mathbf{u}_{i+\frac{1}{2}}^{n+\frac{1}{2}} &= \frac{1}{2}\left[\mathbf{u}_{i+1}^n + \mathbf{u}_i^n\right] - \frac{\Delta t}{2}\left[\frac{\mathbf{f}_{i+1}^n - \mathbf{f}_i^n}{\Delta x}\right] \\ \mathbf{u}_i^{n+1} &= \mathbf{u}_i^n - \Delta t\left[\frac{\mathbf{f}_{i+\frac{1}{2}}^{n+\frac{1}{2}} - \mathbf{f}_{i-\frac{1}{2}}^{n+\frac{1}{2}}}{\Delta x}\right], \end{aligned} \tag{8–4.8}$$

where the values of $\{\mathbf{f}_{i+\frac{1}{2}}^{n+\frac{1}{2}}\}$ in the second step are based on the values $\{\mathbf{u}_{i+\frac{1}{2}}^{n+\frac{1}{2}}\}$ in the first step.

This two-step method can be extended to multidimensions. Consider a two-dimensional generalization of equation (8–4.1),

$$\frac{\partial \mathbf{u}}{\partial t} + \frac{\partial}{\partial x}\mathbf{f}_{\mathbf{x}}(\mathbf{u}) + \frac{\partial}{\partial x}\mathbf{f}_{\mathbf{y}}(\mathbf{u}) = 0, \tag{8–4.9}$$

with the definitions

$$\mathbf{u} \equiv \begin{pmatrix} \rho \\ \rho v_x \\ \rho v_y \\ E \end{pmatrix} \quad \mathbf{f}_{\mathbf{x}} \equiv \begin{pmatrix} \rho v_x \\ P + (\rho v_x)^2/\rho \\ \rho v_x \rho v_y/\rho \\ \rho v_x(E+P)/\rho \end{pmatrix} \quad \mathbf{f}_{\mathbf{y}} \equiv \begin{pmatrix} \rho v_y \\ \rho v_x \rho v_y/\rho \\ P + (\rho v_y)^2/\rho \\ \rho v_y(E+P)/\rho \end{pmatrix}. \tag{8–4.10}$$

A two-step Lax-Wendroff method can be written as two one-dimensional Lax-method steps taking the solution to time $t^{n+\frac{1}{2}}$, and then a leapfrog step:

$$\begin{aligned}
\mathbf{u}^{n+\frac{1}{2}}_{i+\frac{1}{2},j} &= \frac{1}{2}[\mathbf{u}_{i,j} + \mathbf{u}_{i+1,j}]^n \\
&\quad - \frac{\Delta t}{2}\left[\frac{(\mathbf{f}_{\mathbf{x}})_{i+1,j} - (\mathbf{f}_{\mathbf{x}})_{i,j}}{\Delta x} + \frac{(\mathbf{f}_{\mathbf{y}})_{i+\frac{1}{2},j+1} - (\mathbf{f}_{\mathbf{y}})_{i+\frac{1}{2},j-1}}{2\Delta y}\right]^n \\
\mathbf{u}^{n+\frac{1}{2}}_{i,j+\frac{1}{2}} &= \frac{1}{2}[\mathbf{u}_{i,j} + \mathbf{u}_{i,j+1}]^n \\
&\quad - \frac{\Delta t}{2}\left[\frac{(\mathbf{f}_{\mathbf{x}})_{i+1,j+\frac{1}{2}} - (\mathbf{f}_{\mathbf{x}})_{i-1,j+\frac{1}{2}}}{2\Delta x} + \frac{(\mathbf{f}_{\mathbf{y}})_{i,j+1} - (\mathbf{f}_{\mathbf{y}})_{i,j}}{\Delta y}\right]^n \\
\mathbf{u}^{n+1}_{i,j} &= \mathbf{u}^n_{i,j} \\
&\quad - \Delta t\left[\frac{(\mathbf{f}_{\mathbf{x}})_{i+\frac{1}{2},j} - (\mathbf{f}_{\mathbf{x}})_{i-\frac{1}{2},j}}{\Delta x} + \frac{(\mathbf{f}_{\mathbf{y}})_{i,j+\frac{1}{2}} - (\mathbf{f}_{\mathbf{y}})_{i,j-\frac{1}{2}}}{\Delta y}\right]^{n+\frac{1}{2}}.
\end{aligned} \tag{8–4.11}$$

The half-step flux values can be estimated as described in equations (8–1.17) and (8–1.18).

Lax-Wendroff methods are second-order accurate. They give good results for smooth flows with no sharp gradients or discontinuities. Sometimes they can be used when discontinuities are present if optimal spatial resolution and high accuracy at the discontinuity are not required. Such observations apply to any nonmonotone high-order method for solving one or a number of continuity equations, including the compact finite-difference approach described in Section 8–1.4. For general use where discontinuities are present, it is necessary to add artificial diffusion (or artificial viscosity), as discussed briefly in Section 8–1.5. Richtmyer and Morton (1967) and Potter (1973) provide a good introduction to the subject of artificial viscosity. It is also discussed extensively by Hirsch (1990), Anderson (1995), Tannehill et al. (1997). Our general observation is that there are now better ways to treat such flows and to retrofit flux-limiting procedures onto nonmonotone methods.

8–4.2. FCT for Coupled Continuity Equations

We now extend the discussion in Section 8–3 to describe the use of FCT algorithms for integrating coupled continuity equations. The discussion, with perhaps minor changes, also

may be applied to other one-dimensional monotone algorithms. Specifically, consider the three gas-dynamic equations simultaneously,

$$\frac{\partial \rho}{\partial t} = -\nabla \cdot \rho \mathbf{v}, \tag{8–4.12}$$

$$\frac{\partial \rho \mathbf{v}}{\partial t} = -\nabla \cdot (\rho \mathbf{v}\mathbf{v}) - \nabla P \tag{8–4.13}$$

and

$$\frac{\partial E}{\partial t} = -\nabla \cdot E\mathbf{v} - \nabla \cdot (\mathbf{v}P). \tag{8–4.14}$$

We limit the discussion to the one-dimensional problem here, but conclude this section with a description of how to combine several one-dimensional calculations to make a multidimensional monotone calculation.

Solving these coupled equations is done by choosing the timestep, then integrating from the old time t^o for a half timestep to $t^o + \Delta t/2$, and then integrating from t^o to the full timestep $t^o + \Delta t$. The half-step approximation is used to evaluate centered spatial derivatives and fluxes. Assume that the cell-center values of all fluid quantities are known at t^o. The integration procedure is:

1. Half-step calculation to find the first-order time-centered variables.
   a. Calculate $\{v_i^o\}$ and $\{P_i^o\}$ using the old values of $\{\rho_i^o\}$, $\{\rho_i^o v_i^o\}$, and $\{E_i^o\}$.
   b. Convect $\{\rho_i^o\}$ a half timestep to $\{\rho_i^{\frac{1}{2}}\}$.
   c. Evaluate $-\nabla P^o$ for the momentum sources.
   d. Convect $\{\rho_i^o v_i^o\}$ to $\{\rho_i^{\frac{1}{2}} v_i^{\frac{1}{2}}\}$.
   e. Evaluate $-\nabla \cdot (P^o v^o)$ for the energy sources.
   f. Convect $\{E_i^o\}$ to $\{E_i^{\frac{1}{2}}\}$.
2. Whole-step calculation to find second-order results at the end of the timestep.
   a. Calculate $\{v_i^{\frac{1}{2}}\}$ and $\{P_i^{\frac{1}{2}}\}$ using the provisional values $\{\rho_i^{\frac{1}{2}}\}$, $\{\rho_i^{\frac{1}{2}} v_i^{\frac{1}{2}}\}$, and $\{E_i^{\frac{1}{2}}\}$.
   b. Convect $\{\rho_i^o\}$ to $\{\rho_i^1\}$.
   c. Evaluate $-\nabla P^{\frac{1}{2}}$ for the momentum sources.
   d. Convect $\{\rho_i^o v_i^o\}$ to $\{\rho_i^1 v_i^1\}$.
   e. Evaluate $-\nabla \cdot P^{\frac{1}{2}} v^{\frac{1}{2}}$ for the energy sources.
   f. Convect $\{E_i^o\}$ to $\{E_i^1\}$.
3. Do another timestep from t^1 to t^2, as above.

This two-step integration increases the order of the time integration from first to second order. Second-order accuracy in the time integration allows the fourth-order phase accuracy of FCT to have effect.

Often we must couple species continuity equations to the set of equations (8–4.12) through (8–4.14),

$$\frac{\partial n_i}{\partial t} = -\nabla \cdot n_i \mathbf{v}\,, \qquad i = 1, \ldots, N_s. \tag{8–4.15}$$

In general, it is not necessary to integrate these variables during the half step. After

integrating the fluid variables for the half and whole timestep, convect the species concentrations the full timestep using the centered half-step velocities $\{v_i^{\frac{1}{2}}\}$.

There are certain types of errors that arise in the solution of the coupled continuity equations using the timestep splitting, as described above. Phase errors are not completely eliminated, and *synchronizing* the fluxes, so that the flux-correction formula works consistently on the three equations, may be of some concern. Because monotone methods are nonlinear, the amount of flux correction differs from one variable to another. Because a momentum jump may slightly lead the numerical jump in energy in the vicinity of a shock, there is still the possibility of undershoots and overshoots. There are many problem-dependent fixes for these quantitative errors and inaccuracies. One simple cure that often works is to take a smaller timestep. Another is to put a lower limit on derived quantities such as pressure, if such a limit is known. These synchronization inaccuracies are most serious in a chemically reactive flow with a large energy release.

One approach to problems caused by unsynchronized fluxes is to try more complicated ways of limiting the fluxes. When we discussed one-dimensional FCT for a single continuity equation for the generic variable ρ, we described how ρ was flux limited. In general we have the freedom to limit ρ, ρv, and E differently and separately. Various combinations of limiting procedures can help the synchronization problem. Flux limiting and flux synchronization have been discussed by Harten and Zwas (1972) and Zhmakin and Fursenko (1979), and more recently by Zalesak (1997). Note that the methods that incorporate solutions to the Riemann problem (called Riemann solvers) do not have this flux-synchronization issue as it applies to shocks and contact surfaces, because the fluxes given by the Riemann solver, rightly or wrongly, are used as they are. There could, however, be an equivalent issue for subsonic compressible flows where the Riemann solver is not really applicable and flux limiting is still used to avoid generation of unphysical new maxima and minima in the solution. An excellent summary of the use of Riemann solvers in CFD is given by Toro (1997).

8–4.3. Multidimensions through Timestep Splitting

First, rewrite the general continuity equation with source terms as

$$\frac{\partial \varphi}{\partial t} = -\frac{\partial}{\partial x}(\varphi v_x) - \frac{\partial}{\partial y}(\varphi v_y) - \frac{\partial}{\partial z}(\varphi v_z) + \mathcal{S}_x + \mathcal{S}_y + \mathcal{S}_z, \tag{8–4.16}$$

where φ is a generic variable that could be ρ, $\rho \mathbf{v}$, or E. The terms $\mathcal{S}$ represent generic source terms that could appear in continuity equations. Suppose $\varphi = \rho$, the density, then divide equation (8–4.16) into three equations that are advanced sequentially,

$$\begin{aligned}
\frac{\partial \rho}{\partial t} &= -\frac{\partial}{\partial x}(\rho v_x) + \mathcal{S}_x \\
\frac{\partial \rho}{\partial t} &= -\frac{\partial}{\partial y}(\rho v_y) + \mathcal{S}_y \\
\frac{\partial \rho}{\partial t} &= -\frac{\partial}{\partial z}(\rho v_z) + \mathcal{S}_z.
\end{aligned} \tag{8–4.17}$$

In two dimensions, the procedure is as follows:

1. Select a timestep Δt by which the quantity ρ is advanced from ρ^o to ρ^n.
2. Use FCT to solve the x-equation to advance ρ^o by Δt in the x-direction. Define an intermediate value, ρ_x.
3. Use FCT, starting with ρ_x, to solve the y-equation to obtain another intermediate value ρ_{xy}.
4. Set $\rho^n = \rho_{xy}$, and repeat the process.

The solution obtained by this rather straightforward application of one-dimensional continuity-equation solvers often is biased by the ordering of the directions of integration. This can be improved somewhat by varying the order in which the different directions are integrated from timestep to timestep. The most accurate solutions are for cases in which several different permutations are done for one timestep, always starting a series with the same initial value ρ^o, and then averaging the results. Then in two dimensions,

$$\rho^n = \frac{1}{2}(\rho_{xy} + \rho_{yx}). \tag{8–4.18}$$

When this level of averaging is necessary, it is usually more accurate and less expensive to use fully multidimensional algorithms.

One-dimensional solvers for continuity equations can be combined to construct a multidimensional program by timestep splitting in the several coordinate directions. This approach is straightforward when an orthogonal grid can be established with physical boundaries along segments of grid lines. The example shown schematically in Figure 8.5 is a cylindrically symmetric chamber with material entering through a nozzle on the left. The grid is orthogonal and lies along lines of constant r and z. The figure contains a region of high resolution formed by r and z grid lines clustered in the vicinity of the rim of the inlet. In this region, the smallest, most rapidly changing fluid motions arise. Resolving these motions requires high spatial resolution and determines the timestep of the calculation. Other geometries, such as $(x - y)$, $(r - z)$, and the general orthogonal coordinates $(\eta - \xi)$ can also be used with timestep splitting.

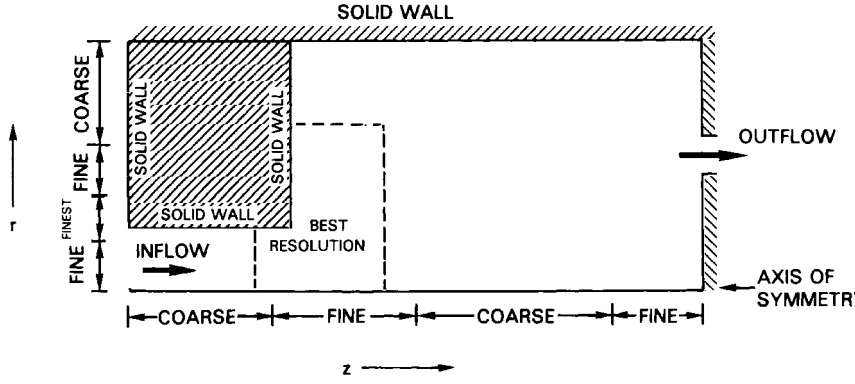

Figure 8.5. Schematic of the grid used in the two-dimensional axisymmetric flow problem.

The two-dimensional equations that describe ideal gas dynamics in axisymmetric $(r-z)$ geometry are

$$\begin{aligned}
\frac{\partial \rho}{\partial t} &= -\frac{\partial}{\partial z}(\rho v_z) - \frac{1}{r}\frac{\partial}{\partial r}(r\rho v_r) \\
\frac{\partial \rho v_r}{\partial t} &= -\frac{\partial}{\partial z}(\rho v_r v_z) - \frac{1}{r}\frac{\partial}{\partial r}(r\rho v_r v_r) - \frac{\partial P}{\partial r} \\
\frac{\partial \rho v_z}{\partial t} &= -\frac{\partial}{\partial z}(\rho v_z v_z) - \frac{1}{r}\frac{\partial}{\partial r}(r\rho v_z v_r) - \frac{\partial P}{\partial z} \\
\frac{\partial E}{\partial t} &= -\frac{\partial}{\partial z}[(E+P)v_z] - \frac{1}{r}\frac{\partial}{\partial r}[r(E+P)v_r].
\end{aligned} \tag{8–4.19}$$

The pressure and energy are again related by

$$E = \epsilon + \frac{1}{2}\rho\left(v_r^2 + v_z^2\right). \tag{8–4.20}$$

The right sides of equation (8–4.19) are separated into two parts, the axial terms and the radial terms. This arrangement in each of the four equations separates the axial and the radial derivatives in the divergence and gradient terms into components that can be treated sequentially by a general one-dimensional continuity equation solver.

Each axial row in the grid is integrated using the one-dimensional module to solve the four coupled continuity equations from time t to $t + \Delta t$. The axial split-step equations are written in the form

$$\begin{aligned}
\frac{\partial \rho}{\partial t} &= -\frac{\partial}{\partial z}(\rho v_z) \\
\frac{\partial \rho v_r}{\partial t} &= -\frac{\partial}{\partial z}(\rho v_r v_z) \\
\frac{\partial \rho v_z}{\partial t} &= -\frac{\partial}{\partial z}(\rho v_z v_z) - \frac{\partial P}{\partial z} \\
\frac{\partial E}{\partial t} &= -\frac{\partial}{\partial z}(E v_z) - \frac{\partial}{\partial z}(P v_z).
\end{aligned} \tag{8–4.21}$$

Because the axial gradients and fluxes are being treated together, the one-dimensional integration connects those cells that are influencing each other through the axial component of convection.

The changes due to the derivatives in the radial direction must now be included. This is done in a second split step of one-dimensional integrations along each radial column,

$$\begin{aligned}
\frac{\partial \rho}{\partial t} &= -\frac{1}{r}\frac{\partial}{\partial r}(r\rho v_r) \\
\frac{\partial \rho v_r}{\partial t} &= -\frac{1}{r}\frac{\partial}{\partial r}(r\rho v_r v_r) - \frac{\partial P}{\partial r} \\
\frac{\partial \rho v_z}{\partial t} &= -\frac{1}{r}\frac{\partial}{\partial r}(r\rho v_z v_r) \\
\frac{\partial E}{\partial t} &= -\frac{1}{r}\frac{\partial}{\partial r}(r E v_r) - \frac{1}{r}\frac{\partial}{\partial r}(r P v_r).
\end{aligned} \tag{8–4.22}$$

The axial and radial integrations are alternated, each pair of sequential integrations constituting a full convection timestep. Thus a single highly optimized algorithm for a reasonably general continuity equation can be used to build up multidimensional fluid dynamics models.

There are obvious limitations to the use of this split-step approach. The timestep must be small enough that the distinct components of the fluxes do not change the properties of a cell appreciably during the timestep. We would like all the terms on the right sides of equation (8–4.22) to be evaluated simultaneously. Their sequential calculation means there is an error. The fractional-step technique used here is second-order accurate as long as the timestep is small and changes slowly enough, but there is still a bias built in depending on which direction is integrated first. To remove this bias, the results from two calculations for each timestep can be averaged, as noted above. This is an expensive, but effective, solution. Chapter 11 contains a more extensive discussion of coupling integrations of the different parts of the reactive-flow equations.

8–4.4. Considerations for Multidimensional Monotone Algorithms

Zalesak (1979) noted that the FCT algorithm could be viewed as six steps performed at each cell or grid point i. Each stage of the implementation of a monotone algorithm is potentially an area where further modifications and variations can be made. We have generalized and recombined Zalesak's six steps (C.R. DeVore, private communication) and now discuss monotone algorithms in terms of the following six stages.

1. Establish a monotone solution at the new time.
2. Determine the extrema, $\rho^{\max}$ and $\rho^{\min}$.
3. Define the correction fluxes including steepening.
4. Ensure monotonicity by directional flux limiting.
5. Limit correction fluxes using multidimensional extrema.
6. Apply limited correction fluxes to obtain the new solution.

These stages can be related to the four steps given at the end of Section 8–3.1 where the basic formulation of one-dimensional FCT was described. Stage 1 corresponds to step 1, and stage 3 corresponds to step 2. The third step in Section 8–3.1 has become stages 2 and 4 here. Stage 5 is new, as it refers to multidimensions. Stage 6 is the same as step 4.

1. Establish a Monotone Solution at the New Time

In stage 1, the algorithm for solving the underlying continuity equation is implemented with sufficient additional diffusion (smoothing) that the provisional solution is monotone at the new timestep. This smoothing can be intrinsic to the algorithm or can be added in a controlled way. As Zalesak and others have shown, the low-order algorithm should be monotone and almost any high-order algorithm can be used as long as enough diffusion is added to suppress purely numerical oscillations. Stage 1 takes the place of the Zalesak first step (computing the transported and diffused fluxes by a low-order method) and includes additional considerations. For example, using Riemann solvers and flux-vector splitting for coupled sets of continuity equations is an attempt to ensure that information

is transferred in the corrct direction and that the numerical solution satisfies the entropy condition (see Chapter 9).

Various different and often complex algorithms may be used to reconstruct profiles of the physical variables to compute more accurate fluxes. For example, even within the direction-split, Lax-Wendroff framework of the standard FCT, there are a number of different averages that can be used to compute the pressure, velocity, and momentum at cell interfaces, given the values at adjacent cell centers. No generally optimal choices have been found; what is best for one problem may not work as well for another. This flexibility is further complicated by the possibility of using different linear combinations of algorithms, as in the hybridization approaches (Section 8–5). At the end of the timestep, the result of stage 1 should be the specification of a provisional solution that is monotone and satisfies the physical constraints and conservation laws of the system.

2. Determine the Extrema, ρ^{max} and ρ^{min}

In stage 2, the solution is inspected to determine the limiting values of the solution variables, which are then used in later stages. Generally, the new value of the convected variable ρ_i in cell i should not exceed the maximum of the surrounding cells (including its own cell), nor should it drop below the minimum value. For a Cartesian three-dimensional grid, taking the maximum and minimum of the surrounding six cell values compared to the local value is a good starting point. In unstructured, irregular, or adaptive mesh grids, the number of neighboring cells will vary. Generally extrema values should be taken with cells that share a face through which a flux may pass.

There are several additional considerations at this stage. Choosing extrema based on previous values in and near that cell is only a convenient approximation to what is happening in the flow. Compression, expansion, and source terms in the fluid-dynamic equations can increase a local maximum value or decrease a local minimum. In addition, convection can bring in a peak from a distance. There have been attempts to incorporate these physical effects directly by using flux-limiters in which the extrema are allowed to change during a timestep. Such limiters can remove much of the clipping observed in strict application of flux correction. They also have the possibly negative effect of reintroducing unphysical overshoots and undershoots near discontinuities and steep gradients. Even without these physics-based criteria for changing the extrema, new maxima and minima still form as they should, but they do so in plateaus about three cells wide.

3. Define the Correction Fluxes Including Steepening

The objective of stage 3 is to define correction fluxes to the provisional solution. For this stage, it is important to know the total amount of diffusion added by the specific algorithms used in stage 1. This information is then used to compute the diffusion fluxes. These diffusion fluxes, minus some amount required for algorithm stability, are candidates for antidiffusive correction fluxes. There are additional important issues related to how these correction fluxes are defined.

One important limit to consider is how the algorithm behaves when the flow velocity reduces to zero. Except for the effects of the limiters described below, the antidiffusive flux between two cells should cancel the diffusive flux exactly when the velocity is zero

in the vicinity of the cell. This has been called the *phoenical* property (Boris, Book, and Hain 1975), and ensures that the solution does not degrade continuously even when there is no flow.

The definition of the correction fluxes includes consideration of steepeners, which should have the effect of further reducing unnecessary smoothing in the final solution by increasing the antidiffusive fluxes. There are at least two reasons to use steepeners. The first is to correct possible problems when the antidiffusive fluxes might smooth instead of steepen the solutions. The second is simply to improve the solution, when this is possible.

Zalesak noticed that sometimes the sign of the density gradient in the provisional solution was opposite to the sign of the antidiffusive flux. As a result, the antidiffusion flux actually smoothed rather than steepened the solution. The original one-dimensional FCT algorithms remedied this in a very simple way, by changing the sign of the raw antidiffusive flux. It was not necessary to change the magnitude since the ensuing flux-limiting stages reduce the magnitude if the steepening is too large. This was done by construction in the original one-dimensional FCT to reduce the operation count, not for purposes of maintaining accuracy. DeVore (1998) extended this steepening modification to multidimensions.

The second reason for steepening is bolder. It is based on the idea that if a little steepening (antidiffusion) is a good thing, then maybe more is better. The idea of steepening was introduced in Section 8–3.3. Nonlinear, monotone solutions can usually be steepened considerably without causing numerical instability. A good flux limiter ensures that shocks and contact surfaces are crisp and convected profiles retain coherence after being convected for millions of timesteps (Cheatham et al. n.d.). Any added steepening increases the amplification factor for some of the intermediate-wavelength harmonics (typically from six to twelve cells in wavelength) of the solution. These wavelengths would otherwise be damped strongly by the combined effects of the underlying algorithm and the nonlinear flux limiter. In the absence of flux limiting, most of these steepeners drive the amplification factor above unity for some portion of the spectrum. This effect may keep contact surfaces sharp, but it can also create unphysical steep gradients where they did not previously exist. These considerations are very much coupled to the form of the limiter in use. When weaker forms of limiting are used, such as those that do not guarantee strict monotonicity (for example, are only "essentially nonoscillatory"), steepening could be disastrous.

4. Ensure Monotonicity by Directional Flux Limiting

Directional flux limiting is used to ensure a stricter form of multidimensional monotonicity than simple application of Zalesak formulas will allow. Situations can frequently arise in time-dependent, high-resolution, multidimensional computations where a flow with steep gradients is closely aligned with the grid or becomes almost stationary with respect to the grid. Figure 8.6 shows one such configuration which could, for example, represent a curved contact surface where the convected gradient is quite large in one grid direction. In the direction perpendicular to the steep gradient, indicated by the arrow, there is a large oscillation in the values in one direction, even though they are monotone in the other direction. Thus the presence of the strong gradient can mask significant unphysical fluctuations. To combat this, the fluxes can be limited separately in the independent grid

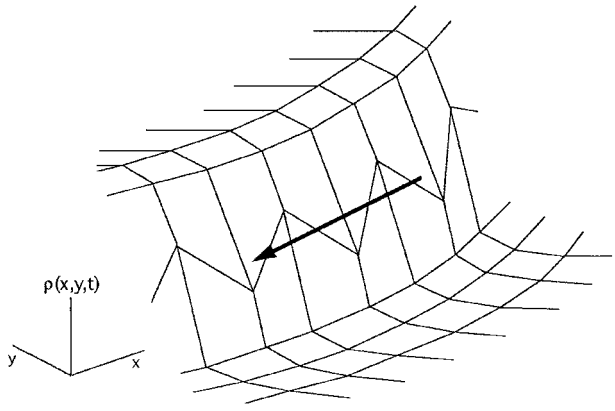

Figure 8.6. Schematic of a situation in which the convected variable ρ has a steep gradient in only one direction. A convection algorithm may produce large oscillations, perpendicular to the gradient, that are not eliminated by flux limiting without special considerations.

directions before the composite limiter is applied. The directional flux limiting is an additional restriction beyond the multidimensional limiting discussed below.

Figure 8.6 also suggests another approach to this problem, which is to add some numerical diffusion perpendicular to the direction of the predominant (averaged) gradient. This is the net effect of directional limiting. To have the desired effect, the gradient used to determine the direction in which to apply the anisotropic diffusion cannot be defined simply as the local gradient. The action of this anisotropic diffusion, added solely for numerical smoothing purposes, must be monitored carefully. As with the use of additional steepeners or locally changing the range of the permissible extrema, such additions can lead to more subtle numerical instabilities that strong flux limiting might otherwise shut down.

5. Limit Correction Fluxes Using Multidimensional Extrema

Stage 5 limits the composite fluxes, a more general and extensive process than ensuring the directional monotonicity considered in stage 4. Fluxes from one direction can destroy monotonicity in a different direction, the major effect ignored in stage 4 and a major difference between multidimensional and direction-split monotone algorithms. For each cell, it is usual to consider separately the fluxes that increase the cell value and those that decrease the cell value. The sum of all the fluxes into the cell is bounded by the extrema computed in stage 3. This concern must be further considered when using adaptive mesh refinement (see Chapter 6), where there is cell merging and there could be sets of fluxes coming from the same direction, but from different neighboring cells.

In a multidimensional problem, there may be several different fluxes entering and leaving the cell at the same time. Therefore multidimensional limiting is usually accomplished by multiplying each flux by a coefficient a_i, ranging between zero and one, such that the new cell value ρ_i just reaches, but does not exceed, the extremum. Unfortunately, there is still a lot of freedom in this computation. For example, there is no good reason, other than simplicity, to reduce each flux by the same factor to ensure an extremum limit is maintained. The fluxes could be reduced in proportion to the volume of the cells from

which they originate, in proportion to the relative cell interface areas across which each of the fluxes penetrates the cell, or according to some other prescription.

Attempts to improve the simpler, more conservative, flux-limiting formulations are more expensive and may be dangerous. For example, flux limiters usually treat incoming and outgoing fluxes separately. One result of this is that a set of antidiffusive fluxes may be reduced by one of the limiters even though the action of the fluxes of opposite sign will be sufficient to prevent an extremum being exceeded. It would appear that reducing the action of the limiter on one set of fluxes in anticipation of fluxes of the opposite sign would reduce the net effect of the limiter and allow steeper solutions. One or more of the fluxes of opposite sign, that are being used to reduce the effect of the limiter, may themselves be limited. This means that there might be less flux cancellation than expected, which creates the need for a complicated, iterative procedure to ensure that the extrema are respected. A number of attempts have been made along these lines with little success. Methods that are better in one case are worse in others, or they are appreciably more complicated and expensive for the demonstrable gains made.

In addition to combining the effects of fluxes from all different directions, some implementations consider the simultaneous, interactive action of limiting several variables at once. Aspects of this multivariable limiting are sometimes referred to as *flux synchronization*. Unsynchronized fluxes may be viewed as a problem or as an opportunity. For example, if the monotone algorithm moves too much momentum into a cell compared to the amount of energy it has moved, the kinetic energy is too large, forcing the temperature to be too low. Flux synchronization, the attempt to correct this type of problem, is a flexibility afforded by the FCT, and one that is absent by construction in Gudonov methods and related techniques using Riemann solvers. The most rigorous synchronization usually means using the same antidiffusion coefficient for all variables. This results in a solution that is synchronized, but much too diffusive compared with the solutions based on independently limiting the individual continuity equations.

6. Apply Limited Correction Fluxes to Obtain the New Solution

Stage 6 completes the computation for the timestep. The limited, antidiffusive fluxes are added to the provisional (monotone) solution, and the results are used to update all of the dependent physical variables. The smoothing that was introduced to ensure monotonicity has been reduced to the lowest level permissible, as determined by the combined action of the limiters and the extrema computations. It is important to keep in mind that the definition of what is "permissible" is subjective because almost every algorithmic decision involves some form of trade-off.

8–4.5. Flux-Corrected Transport Packages

The earliest FCT package (Boris 1976) has recently been expanded and generalized into LCPFCT, which is a particular implementation of FCT designed to solve a generalized continuity equation,

$$\frac{\partial \rho}{\partial t} = -\frac{1}{r^{\alpha-1}}\frac{\partial}{\partial r}(r^{\alpha-1}\rho v) + \frac{1}{r^{\alpha-1}}\frac{\partial}{\partial r}(r^{\alpha-1}D_1) + C_2\frac{\partial D_2}{\partial r} + D_3. \tag{8–4.23}$$

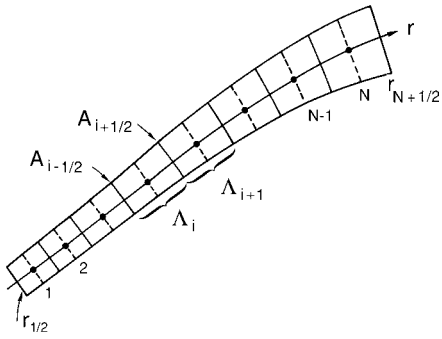

Figure 8.7. Geometry for the LCPFCT grid showing cell and interface numbering and defining geometric grid variables.

LCPFCT may be used to solve sets of coupled continuity equations (Boris et al. 1993). This formulation allows the possibility of a variable and moving grid, additional source terms (D_1, D_2, and D_3), and variable, one-dimensional geometries through the variable α ($\alpha = 1$ is Cartesian or planar, $\alpha = 2$ is cylindrical, and $\alpha = 3$ is spherical geometry). The package synthesizes a great deal of the information accumulated in the past twenty years on the best ways to solve continuity equations. Many of the subtleties involved in using LCPFCT successfully, including various ways to improve speed and performance, are discussed in Boris et al. (1993), which includes the LCPFCT suite of subroutines, test programs, and extensive documentation. Considering the complexity of LCPFCT compared to the other convection algorithms given in Chapter 4, it is surprising that it only requires twice the computational time.

Figure 8.7 shows a one-dimensional geometry in which the fluid is constrained to move along a tube. Here the variable r measures length along the tube, and the velocity v^f is the fluid velocity along r. The points at the interfaces between cells are the finite-difference grid points. The interface positions at the beginning of a numerical timestep are denoted by $\{r^o_{i+\frac{1}{2}}\}$, where $i = 0,\ 1, \ldots, N$. At the end of a timestep Δt, the interfaces are at $\{r^n_{i+\frac{1}{2}}\}$, where

$$r^n_{i+\frac{1}{2}} = r^o_{i+\frac{1}{2}} + v^g_{i+\frac{1}{2}}\,\Delta t. \tag{8–4.24}$$

Here the quantities $\{v^g_{i+\frac{1}{2}}\}$ are the velocities of the cell interfaces. Figure 8.7 also indicates the basic cell volumes $\{\Lambda_i\}$, and the interface areas $\{A_{i+\frac{1}{2}}\}$. The interface areas are assumed to be perpendicular to the tube and hence to the velocities $\{v^f_{i+\frac{1}{2}}\}$. The change in the total amount of a convected quantity in a cell is the algebraic sum of the fluxes of that quantity into and out of the cell through the interfaces. Both the cell volumes $\{\Lambda_i\}$ and the interface areas $\{A_{i+\frac{1}{2}}\}$ that bound these cells have to be calculated consistently using new and old grid positions.

The positions of the cell centers are denoted $\{r^{o,n}_i\}$ and are related to the cell interface locations by

$$r^{o,n}_i = \frac{1}{2}\left[r^{o,n}_{i+\frac{1}{2}} + r^{o,n}_{i-\frac{1}{2}}\right], \qquad i = 1,\ 2\ , \ldots, N. \tag{8–4.25}$$

The superscripts o and n indicate the old and new grid at the beginning and the end of the timestep. The cell centers could also be computed as some weighted average of the

interface locations and are generally only used for calculating diffusion terms added to equation (8–4.23). The boundary interface positions defining the grid, $r^{o,n}_{\frac{1}{2}}$ and $r^{o,n}_{N+\frac{1}{2}}$, have to be specified by the user. For example, they might be the location of bounding walls. The effect of a piston or flexible container can be simulated by programming $r_{\frac{1}{2}}$ as a function of time and forcing the adjacent grid points to move correspondingly.

To calculate convection, determine the flux of fluid through the interface as it moves from $r^o_{i+\frac{1}{2}}$ to $r^n_{i+\frac{1}{2}}$ during a timestep. The velocities of the fluid are assumed known at the cell centers and the velocity of the fluid at the interfaces is approximated by

$$v^f_{i+\frac{1}{2}} = \frac{1}{2}\left(v^f_{i+1} + v^f_i\right), \qquad i = 1,\ 2,\ \ldots, N-1. \tag{8–4.26}$$

Again, other weighted averages are possible but this choice works well for all three geometries.

The fluxes out of one cell and into the next are needed on the interfaces. Define

$$\Delta v_{i+\frac{1}{2}} = v^f_{i+\frac{1}{2}} - v^g_{i+\frac{1}{2}}, \qquad i = 1, 2, \ldots, N-1. \tag{8–4.27}$$

In this formulation, the only velocity that matters is the relative velocity between the fluid and the interface, which may be moving. This means that the model may be made Lagrangian by moving the grid at the local fluid velocity. This simple approach to a Lagrangian formulation is not wholly satisfactory for several reasons that are discussed in Chapter 9.

LCPFCT vectorizes and parallelizes in a straightforward way, requiring approximately thirty floating-point operations per continuity equation per timestep per spatial direction. It has been programmed in various versions of C and Fortran, and optimized for many types of vector and parallel computers. LCPFCT has been incorporated in the fully scalable applications program FAST3D, which runs on a number of parallel computers and uses the VCE algorithm described in Chapter 6 to compute flows with complex boundaries (Landsberg et al. 1993; Landsberg and Boris 1997).

Another FCT package provides a method for solving the generalized continuity and hydromagnetic equations in two dimensions. (This package is available from the NASA HPCC/ESS software exchange. Contact: C.R. DeVore at the U.S. Naval Research Laboratory.) The FCT algorithms in this package are standard in the sense that they are conservative and use a finite-volume representation. There is considerable flexibility in the geometries and boundary conditions that may be simulated. There are versions for several supercomputers and parallel processors. One group of models using these multidimensional FCT routines exploits the positivity- and monotonicity-preserving properties to obtain solutions of the compressible, ideal MHD equations. Data-parallel implementations of the latest multidimensional FCT algorithms have been developed for fluid dynamics and MHD in two and three dimensions.

There are several reactive Navier-Stokes solvers built on FCT that are not generally available, but they show the broad range of uses of the algorithm. For example, the vectorized program developed by Vuillermoz and Oran (1992) was used to study turbulent supersonic reacting flows, and the data parallel program developed by Weber et al. (1995, 1997) was used to study the interaction of shocks and boundary layers in endothermic

gases and detonation cell structure (Oran et al. 1998). In addition to the FCT algorithm for fluid dynamics and representations of viscous diffusion and thermal conduction, these packages incorporate multispecies, multireaction chemical-kinetics mechanisms. In addition, a number of flame models were developed around the implicit BIC-FCT algorithms introduced in Section 8–3.2 for slow reactive flows. These are discussed further in Chapter 9 and include two- and three-dimensional flame models, FLAME2D and FLAME3D (Patnaik et al. 1989; Patnaik and Kailasanath 1996; Ellzey, Laskey, and Oran 1991; Kaplan et al. 1994). In addition to the standard viscous diffusion terms, these models also include representations for radiation transport. All of these programs have been parallelized on a number of different computers (for example, Moon et al. [1995]).

8–5. Expansion Methods

Expansion methods express fluid variables as linear combinations of basis functions whose coefficients are determined by integrating a complicated set of coupled, usually nonlinear, ordinary differential equations in time. For finite elements, the basis functions are usually piecewise polynomials localized to a few computational cells. In contrast, spectral and wavelet methods use global basis functions. Spectral element methods, a combination of both finite-element and spectral ideas, use a number of simple, localized, contiguous subdomains within which expansions of spectral accuracy are used.

This section is a brief overview of some of the features of expansion methods, emphasizing their similarities and differences, and giving some indication of their current use and applicability for reactive flows. None of the methods is presented in the detail used to describe finite-volume methods. Of the expansion methods, finite-element methods are the most used in fluid dynamics problems. They have been particularly useful for problems with separating boundary-layers flows and multidisciplinary problems involving coupling of fluid and structural dynamics. Their major drawback is the complexity of the programming and cost in terms of memory and computer time. There are fundamental problems with applying expansion methods to reactive flows, especially when each expansion function spans a wide domain. We will try to explain these issues in the material presented in Section 8–5.1, where we also indicate areas of current active research.

8–5.1. Galerkin, Tau, and Collocation Approximations

Three basic techniques are used in finite-element and spectral methods: Galerkin, tau, and collocation approximations. To understand how these are defined and differ, consider an initial value problem of the form

$$\frac{\partial u(x,t)}{\partial t} = \mathcal{L}(x,t)[u(x,t)] + f(x,t). \tag{8–5.1}$$

Here $\mathcal{L}$ is generally a nonlinear operator. For our current interests, u may be one of the dependent variables described by the continuity equation, such as ρ, ρv, or E, and $\mathcal{L}$ may be the spatial derivative operator in the continuity equation discussed in Section 8–1.

Assume that boundary conditions of the form

$$B_{\pm}u(x,t) = 0 \tag{8–5.2}$$

are applied at the boundaries $x = x_{\pm}$, where $B_{\pm}$ are linear operators. Define the initial conditions as

$$u(x,0) = u_o(x). \tag{8–5.3}$$

Approximate the function u by a superposition of N basis functions $\{v_n(x)\}$, each satisfying the boundary condition in equation (8–5.2),

$$u(x,t) \equiv \sum_{n=1}^{N} a_n(t)v_n(x). \tag{8–5.4}$$

If the expansion functions $\{v_n(x)\}$ form an orthonormal set, the coefficients $\{a_n\}$ satisfy

$$a_n(t) = \int_{x_-}^{x_+} dx\ v_n(x)u(x,t)w(x), \tag{8–5.5}$$

where $w(x)$ is a suitable weighting function assumed here to be unity. Equation (8–5.1) gives evolution equations for these coefficients,

$$\dot{a}_n \equiv \frac{da_n}{dt} = \sum_{m=1}^{N} \int_{x_-}^{x_+} dx\ v_n(x)\mathcal{L}v_m(x) + \eta_n(t), \tag{8–5.6}$$

where $\{\eta_n\}$, the expansion coefficients of the inhomogeneous term in equation (8–5.1), are chosen to satisfy

$$f(x,t) = \sum_{n=1}^{N} \eta_n(t)v_n(x). \tag{8–5.7}$$

Equation (8–5.6) is an explicit equation for $\dot{a}_n$. It has this particularly uncomplicated form because we have assumed that $\{v_n\}$ are orthonormal and the boundary condition operators $B_{\pm}$ do not change in time. In a more general case, it is necessary to solve N linear equations simultaneously,

$$\begin{aligned} \sum_{m=1}^{N} \left\{ \dot{a}_m \int_{x_-}^{x_+} dx\ v_n(x)v_m(x) - \int_{x_-}^{x_+} dx\ v_n(x)\mathcal{L}v_m(x) \right\} \\ = \sum_{n=1}^{N} \eta_m \int_{x_-}^{x_+} dx\ v_n v_m, \qquad n = 1, 2, \ldots, N. \end{aligned} \tag{8–5.8}$$

Together, equations (8–5.4) and (8–5.8) are the *Galerkin* approximation. A review of the application of Galerkin methods has been given by Fletcher (1984).

The *tau* approximation (Lanczos 1956) also starts with equation (8–5.4), but the functions $v_n(x)$ do not have to satisfy the boundary conditions. Instead, there are two additional

constraints,

$$\sum_{n=1}^{N} a_n B_{\pm} v_n(x) = 0. \tag{8–5.9}$$

The solution is found from these two equations along with the first $N-2$ of equation (8–5.8) or (8–5.6).

In *collocation* methods, N points $\{x_n\}$ are chosen in the range (x_-, x_+), such that

$$x_- < x_1 < \cdots < x_N < x_+. \tag{8–5.10}$$

These collocation points are effectively the grid points of the method. The solution for $u(x)$ in equation (8–5.1) is approximated by equation (8–5.4), so that the coefficients a_n satisfy

$$\sum_{m=1}^{N} a_m v_m(x_n) = u(x_n)\,, \qquad n = 1, 2, \ldots, N. \tag{8–5.11}$$

For example, if $\{x_n\}$ are uniformly distributed between x_- and x_+, and $\{v_n\}$ are functions such as sines and cosines, equation (8–5.11) is a discrete Fourier transform. The prescription for collocation methods is then to use equation (8–5.4) in equation (8–5.1), and find a set of N equations for $\{a_n\}$ by evaluating equation (8–5.1) at the points $\{x_n\}$. The distinguishing feature is that the continuum equation is satisfied exactly at only the grid points.

In all three of these approximations, we have not yet specified the $\{v_n\}$ or supplied a prescription for carrying out the time integration. In each case, the formulation converts equation (8–5.1) into a set of ordinary differential equations.

8–5.2. Spectral Methods

In spectral methods, the dependent variables are written as linear combinations of a set of basis functions of the independent variables, as in equation (8–5.4). The basis functions are constructed with properties that simplify calculation of the series coefficients. For example, the functions should be analytic and orthogonal, should have some connection with the physical processes in the problem, and, in some cases, should satisfy boundary conditions. Both spectral methods and finite-element methods (discussed later in this section) can be based on any of the three approximations described in Section 8–5.1: Galerkin, tau, or collocation. A number of different spectral approximations are possible, depending on which technique is adopted, the choice of basis functions, and the order of the approximation. All three types of spectral techniques can be generalized in a straightforward way to multidimensions. The time derivative is generally finite differenced and integrated either explicitly or implicitly. For example, an explicit integration might difference time by a leapfrog or a Runge-Kutta method (see Chapter 5).

Early references for spectral methods include Orszag (1972), Kreiss and Oliger (1973), the collection of papers in Voigt, Gottlieb, and Hussaini (1984), and the book by Hussaini and Zang (1984). Gottlieb and Orszag (1977) considered finite-difference time integrations for spectral methods and Orszag (1980) considered how to treat complex geometries using

spectral algorithms. This technology is necessary for realistic engineering problems and leads to the spectral-element methods discussed below.

Aliasing errors are encountered in conjunction with spectral methods. There is a very clear discussion of these errors in Roache (1982). The shortest-wavelength component of the expansion of $u(x, t)$ that can be resolved in a computational mesh has $\lambda = 2\Delta x$. Because there is generally more interest in accuracy at long wavelengths, this short-wavelength cut-off might not seem very important. Short wavelengths are important, however, if they interfere with the long wavelengths. In many situations, the spectral components of u interact in such a way that too much energy appears in the long wavelengths, energy that should appear in the much shorter, unresolved wavelengths. On the coarse grid, these omitted short-wavelength components can appear as long *aliased* wavelengths.

When the Galerkin approximation is used, the method is commonly called *spectral* or *fully spectral*. The basis sets used are generally orthonormal. When they are not, the integrals must be computed numerically. If the basis functions are orthonormal, and the operator $\mathcal{L}$ is linear, it is not necessary to solve a matrix for the $\{a_n\}$. If $\mathcal{L}$ is nonlinear, for some orthonormal bases there are extremely fast convolution processes which make the evaluation of the nonlinear terms relatively inexpensive. Such bases include, for example, Fourier and Chebyshev expansions, for which fast Fourier transforms and real cosine transforms exist. Aliasing errors do not appear in Galerkin and tau spectral methods, which throw away the terms with extraneous frequencies generated in the nonlinear terms. That is, they eliminate the spurious frequencies from the configuration-space interactions. Formally, they enforce the condition that the error in the approximating function is orthogonal to the expansion functions. Generally, tau and Galerkin methods require twice the computational work of the collocation methods.

A *pseudospectral* method uses a collocation of points $\{x_i\}$ chosen to make the evaluation of equation (8–5.6) very efficient. For example, if Fourier series are used for the basis, the $\{x_i\}$ are equally spaced. If Chebyshev polynomials are used, the $\{x_i\}$ are roots of the polynomials. In these two representations, it is not necessary to solve a matrix, even when $\mathcal{L}$ is nonlinear. Pseudospectral methods contain aliasing errors unless these are independently filtered. The aliasing errors are not large and can be neglected in many cases with enough dissipation in the calculation. Combining a pseudospectral method with suitable filtering essentially gives a spectral method.

For spectral methods to be competitive with finite-difference and finite-volume methods, very efficient algorithms are needed for performing the transform, equation (8–5.5), and its inverse, equation (8–5.4). Both of these transformations generally require N^2 operations to perform, where N is the number of basis functions. Fortunately, there are fast Fourier transform (FFT) algorithms whose cost scales as $N \log N$ rather than N^2. These very fast methods have been developed as standard subroutines for many kinds of computer platforms. Furthermore, Orszag (1972) showed that pseudospectral approximations can be about as accurate as Galerkin approximations at much less cost.

Consider an application of spectral methods to the two-dimensional incompressible Euler or Navier-Stokes equations (Gottlieb and Orszag 1977). These methods are often solved in the vorticity and stream function formulation. The difficulties in solving the equations arise from nonlinear terms in the $\mathcal{L}$ operator in equation (8–5.6). The key is to use the fast transform to evaluate the derivative $\partial u/\partial x$, where now u denotes the vorticity

ω, and to evaluate the equation in conservation form in physical space. For complex exponential basis functions,

$$\frac{\partial u}{\partial x} = \frac{2\pi}{x_+ - x_-} \sum_{n=1}^{N} i\, n a_n \exp\left(\frac{2\pi i n \Delta x}{x_+ - x_-}\right). \tag{8–5.12}$$

The solution is then integrated forward in time at the collocation points in configuration space.

Thus it appears that pseudospectral approximations have several advantages over Galerkin methods. They have fewer FFTs per step, or the same number on a smaller grid; they are simpler, and have less difficulty with complicated boundary conditions. Galerkin methods have no aliasing errors, and, as a wavenumber representation, spectra are readily and accurately available for diagnostic interpretations. Even when FFTs are used, over 80 percent of the time required by a spectral calculation is spent doing transforms whose cost can become excessive for physically complex, nonlinear problems.

When the problem has periodic boundary conditions, the basis should be composed of complex Fourier functions. If sinusoidal functions are used for problems without periodic boundary conditions, the answers can be relatively poor throughout the entire domain. This is understandable in terms of the global nature of spectral methods. All of the points are used to compute derivatives so an error at one location is felt immediately all over the computational region. When the boundary conditions are not periodic, the basis set should consist of, for example, Chebyshev or Legendre polynomials. Chebyshev polynomials are useful not only because they allow a variety of boundary conditions, but because FFTs (in particular, real cosine transforms) can be used very efficiently with them.

There have been efforts to extend spectral methods to compressible flows (starting with Gottlieb and Orszag [1977], Gottlieb, Hussaini, and Orszag [1984], Hussaini and Zang [1984]). There is a fundamental concern with this approach. The method is intrinsically global so the entire computational domain instantly senses what is happening everywhere else. This means that local phenomena such as shocks respond to what happens outside the Mach cone, a physically unrealistic situation. Implicit methods have this same problem with shocks, but the problem exists for both implicit and explicit spectral methods. Two examples of problems that arise are shocks with large oscillations from the Gibbs phenomena and numerical precursors in the material ahead of the shock. Possible remedies are to filter the unwanted information from the region around the discontinuity or to use shock-fitting techniques.

Spectral algorithms do not guarantee positivity in the way that monotone algorithms do. Therefore, in their simplest form, they are not as good for resolving moving discontinuities such as shock fronts or breaking internal waves. There have been several attempts to incorporate monotonicity in spectral methods through flux-limiting procedures, such as the initial effort of Boris and Book (1976b), Taylor, Myers, and Albert (1981), Zalesak (1981), and McDonald et al. (1985). As long as there are no sharp gradients and the boundaries are relatively simple, spectral methods require less resolution than finite-difference methods as they have, in a sense, infinite-order accuracy. Thus they are competitive with, and sometimes superior to, finite-difference methods for incompressible flows.

Spectral methods became extremely popular with the development of fast transform methods and have been used extensively in numerical weather prediction and simulations

of certain types of idealized turbulent flows (discussed in Chapter 12). In fact, these methods are best for idealized flows with only a few nonlinear terms where smoothly varying solutions must be very accurate. For idealized shock problems, filters can be found and the calculations appear satisfactory. The methods are not as useful for problems involving discontinuities, complex shock interactions, or complex geometries. Thus they are not generally used for engineering problems. Spectral element methods, Section 8–5.4, are an approach for bypassing the limitations imposed by the global and nonmonotone properties of spectral methods.

8–5.3. Finite-Element Methods

Finite-element techniques have been in use for decades to solve difficult, practical problems in structural engineering. Mathematicians subsequently formulated their properties in terms of broad classes of approximations. Since then, finite-element methods have been developed for a variety of problems in fluid dynamics and heat transfer, and especially now for time-dependent fluid dynamics. Their application to solutions of systems of continuity equations are of interest here. Good general references to finite-element methods include Strang and Fix (1973) and Zienkiewicz (1977). The book by Zienkiewicz and Morgan (1983) introduces the use of finite elements for solving partial differential equations. An excellent, comprehensive text on the finite-element method showing its broad use in engineering problems is the two-volume set by Zienkiewicz and Taylor (1989, 1991). There are a number of books and compendia containing some of the latest innovations and applications of finite elements to fluid dynamics. An extensive review of the use of unstructured grids in conjunction with finite-element methods has been given by Morgan and Peraire (1998).

There are two ways of viewing finite-element methods for solving partial differential equations. The first is global. The independent variable u is expanded over the whole region (x_-, x_+), as written in equation (8–5.4). Here N is the number of functions used in the expansion. One common choice for the basis functions v_n is called a *tent*, *roof*, or *teepee* function,

$$v_n(x) = \begin{cases} \dfrac{x - x_{n-1}}{x_n - x_{n-1}} & \text{if } x_{n-1} \le x \le x_n, \\ \dfrac{x_{n+1} - x}{x_{n+1} - x_n} & \text{if } x_n \le x \le x_{n+1}, \\ 0 & \text{otherwise,} \end{cases} \tag{8–5.13}$$

where the $\{x_n\}$ satisfy equation (8–5.10). In this representation, each v_n consists of two straight lines. One of the lines has a positive slope between the *nodes* x_{n-1} and x_n and the other line has a negative slope between the nodes x_n and x_{n+1}. The region between two neighboring nodes is called an *element*. Equation (8–5.13) defines the *shape function* that spans the subregion $[x_{n-1}, x_{n+1}]$. This piecewise linear shape function has nonzero values over two elements, and is zero elsewhere, as discussed in Chapter 6.

Many choices of v_n are possible. The number of functions needed in an expansion such as equation (8–5.4) is related to the degree of the polynomial, the smoothness or continuity imposed on the nodes, and the number of elements in the domain.

The second approach is local. We expand each element in a polynomial

$$v_e(x) = \sum_{i=1}^{I} a_{e,i} x^i \tag{8–5.14}$$

so that the coefficient $a_{e,i}$ refers now to the coefficient of the ith term of the polynomial for element e. In the global representation, continuity at the nodes is built in. With this local representation, added constraints ensure the proper degree of continuity of the shape functions at the nodes.

The next step in the development of the method is somewhat different from the spectral methods described above. We define an *error function* or *residual function* R_u, as a measure of how much the chosen expansion fails to satisfy the original equation, equation (8–5.1),

$$R_u(\{v_e\}, t) = \frac{\partial u}{\partial t} - \mathcal{L}[u] - f. \tag{8–5.15}$$

Then we ask that

$$\int_{x_-}^{x_+} W_l R_u \, dx = 0, \qquad l = 1, 2, \ldots, N, \tag{8–5.16}$$

where $\{W_l\}$ is a set of independent *weighting functions*. Substituting the expression for u in equation (8–5.4) in equation (8–5.16) leads to a set of coupled ordinary differential equations of the form

$$\mathbf{C} \cdot \dot{\mathbf{a}} + \mathbf{K} \cdot \mathbf{a} - \mathbf{g} = 0. \tag{8–5.17}$$

The $\mathbf{C}$, $\mathbf{K}$, and $\mathbf{g}$ matrices are integrals over the entire region $[x_-, x_+]$ if we are using global functions and over a subregion if we use local functions,

$$\begin{aligned} C_{lm} &= \int W_l N_m \, d\Omega \\ K_{lm} &= -\int W_l \mathcal{L} N_m \, d\Omega \\ g_l &= -\int f W_l \, d\Omega. \end{aligned} \tag{8–5.18}$$

This is called the *weighted residual approximation*. The functions $\{W_l\}$ may be chosen in a number of ways. The choice $W_l = v_l$ gives a Galerkin method. The choice $W_l = \delta(x - x_l)$ gives a point collocation method. Another weighting, $W_l = x^{l-1}$, is called the *method of moments*. Each of these choices has different strengths and weaknesses.

Finite-element methods in multidimensions usually require a great deal of computer storage. If time is advanced by an explicit finite-difference formula, it is necessary to solve a linear matrix problem at each step. When the time is advanced implicitly, there is an additional price in computer time for large matrix inversions at each timestep. Because the cost of recomputing geometric and shape functions can be prohibitive for each physical variable transported, these variables are usually stored, requiring dozens of quantities per node in two dimensions and many more in three dimensions. These costs are becoming less

important for moderate-sized problems as computer speed and memory have increased. Unfortunately, the size of the problems we wish to compute has also increased greatly because of the availability of parallel processing. Another issue is the complexity of the programming itself, and this leads to a relatively long start-up time for those who must do more than run an existing code for a pretested set of input parameters. Finite-element approaches based on matrix inversion are also relatively less efficient on parallel-processing computers than on single processors.

There are strong compensating advantages to finite-element methods that balance the disadvantages of computational and memory costs, and these advantages explain their continuing use in fluid-engineering problems. Perhaps the most obvious is the ability of these methods to use unstructured grids that may be constructed from triangular, quadrilateral, or any polyhedral shape. Such grids and their connection to finite elements were mentioned previously in Chapter 6. Finite elements have become so associated with complex gridding and grid generators that, in an increasing number of cases, the finite-element framework may be used to generate the grid, and then a finite-volume method is actually used to solve the equations. An advantage of using finite-elements is that they may be combined with materials-structure programs to solve problems involving fluid-structure interactions. This is an area of active research. Since finite elements are a mainstay in structural mechanics, coupling a finite-element representation of the fluid dynamics to the structural mechanics seems a natural way to do this difficult, multidisciplinary problem.

In some cases, finite-difference or finite-volume methods can be expressed as a special case of the weighted residual process with locally defined shape functions. Simple finite-difference methods and finite-element methods result in the same algorithms. Although the answer is "generally yes" (see, for example, Zienkiewicz and Morgan [1983]), it takes some effort to provide a proof for individual cases. The finite-volume expressions can be interpreted as a particular case of weighted residuals in which the weights are delta functions, $W_l = \delta(\mathbf{x} - \mathbf{x_l})$, and the shape function represents the derivative in the same way as the finite-difference operator. And this leads to another reason why finite-element methods are successful: they essentially reduce to a finite-volume approach in the limit of the lowest-order expansions in elements. Many of the largest, most complex codes are in fact using some form of finite-volume methods. For this reason, many of the most useful features of finite volumes, such as flux limiting, may be used. For example, Löhner et al. (1988) first applied FCT flux-limiting to a finite-element method. Later work has tested and applied a variety of other limiters and considered many variations of the algorithms, (see Luo, Baum, and Löhner [1994]).

8–5.4. Spectral-Element Methods

Between highly localized and global representations, there are families of expansions that become progressively more delocalized in the sense that they use more values from cells that are further away. One such intermediate representation is called *spectral elements*, in which the overall computational domain is divided into a number of regular subdomains, called *elements*. These methods were developed to combine the high-order accuracy advantages of spectral methods with the ability of block-structured grids to represent complex

body shapes. As discussed previously in this chapter, localized representations are better suited to discontinuous phenomena since they more easily reproduce physically correct finite information propagation speeds.

An important part of the spectral-element method is that the degrees of freedom of the expansion in a high-order polynomial are mapped onto the variably shaped elements in the computational domain. Thus spectral-elements are rather like finite elements, except now the expansion functions are based on the Lagrange interpolants with the nodes being unequally distributed according to the Chebyshev spacing. In this approach, the treatment within each element is essentially spectral.

Spectral elements were first successfully applied to incompressible flows (Patera 1984; Canuto et al. 1988). Because this approach can be applied on unstructured grids, it has a definite advantage with respect to spectral methods when the boundaries are complex. Spectral-element methods have now been expanded to treat compressible flows by enforcing monotonicity and positivity through flux-limiting procedures (Giannakouros and Karniadakis 1994; Lomtev, Quillen, and Karniadakis 1998). An excellent book by Karniadakis (1998) describes and extends the method.

8–5.5. Wavelet Methods

Because of the global nature of the expansion functions in, for example, Fourier series, no terms in the expansion can be safely ignored. This is a fundamental problem in using spectral methods generally for fluid dynamics. When there are shocks or discontinuities, a local analysis generally gives a much better representation. There have been substantial efforts to extend these nonlocal expansion methods to handle shock and discontinuity problems.

Wavelet approaches are based on an expansion that uses basis sets that are better suited to representing local structures in the solution than standard spectral methods. To date, wavelets have been applied to signal processing, although now there is a notable effort to extend the applications to solutions of partial differential equations. Rather than expanding within spatially separate blocks, as in spectral-element methods, an expansion is used in which there are a number of expansion functions for each region, each representing a different scale length in the solution. Again, the expansion functions depend on both k and x. Unlike spectral elements, the expansion functions are not grouped in patches and may potentially be used fully adaptively. There are conditions, restrictions, and optimal choices of the basis set for different types of problems.

A good introduction to wavelets has been given in the book by Benedetto and Frazier (1994), a compendium of articles on various aspects and applications of wavelets. More recent work of Beylkin and Keiser (1997) has considered extensions to partial differential equations and fluid dynamics. These methods are in an early stage of development. Issues such as monotonicity and positivity have not yet been addressed, although the same kinds of considerations must eventually be brought up. A potential advantage of wavelets is the ability to adjust the basis set, since it is not fully determined *a priori*, to satisfy additional constraints or properties of the solution. To date, it is not clear that wavelet methods will be as generally useful in fluid dynamics as they are in signal processing, but this is an area of current research. Adaptive mesh refinement in configuration space would seem to accomplish the same ends as adaptive wavelets in wavenumber space.

8–6. Resolution and Reynolds-Number Limitations

The actual fluid dynamics equations (see Chapter 2) describe not only convection, but also other physical processes such as viscosity and thermal conduction. The result is the Navier-Stokes equations. The additional physical complexity not only complicates the solution procedure, but also provides opportunities for improved solution methods. To close this chapter, we discuss some of the limitations of numerical resolution. We show that these limitations cannot be separated from properties of the fluid physics, especially the viscosity.

In Chapter 2 we defined the nondimensional Reynolds number,

$$Re \equiv \frac{Lv}{\nu} , \tag{8–6.1}$$

as the ratio of the viscous (diffusive) decay time of a flow structure, with characteristic dimension L, to the transit time of fluid across that structure based on the characteristic velocity v. The Reynolds number is often used as a guide for making qualitative judgments about the flow, such as whether it is dominated by viscous damping or whether turbulence can develop.

Because a fluid flow often involves several space and velocity scales, choosing the values that go into the definition of the Reynolds number is somewhat ambiguous. The long-wavelength components of the flow are not strongly damped because the viscosity is small. Nonetheless, if the flow has persisted long enough, the short-wavelength components are either gone or are very small. This suggests that a complicated flow might be characterized by several different Reynolds numbers. For example, two Reynolds numbers might be estimated, one for the short-wavelength structures and one for the long-wavelength structures. Different velocities as well as scale lengths may be appropriate. Because most interesting flows are complicated, this separation can never be perfect and somewhat limits the usefulness of nondimensional analysis.

The issues of resolution and numerical diffusion in simulating convective flows on an Eulerian grid are closely intertwined. The flow generally has structure on many different spatial scales simultaneously. For each application, there can often be disagreement about how well the various scales have to be resolved in each region of the computational domain. In some cases, it is possible to resolve only the range of scales needed and not be concerned about others. In other cases, the flow has important structures on any spatial scale we can afford to simulate numerically.

The least-damped flow that a simple Eulerian method can describe may be characterized by a Reynolds number calculated as if the numerical diffusion were actually the physical diffusion. We can evaluate this Reynolds number for FCT using the dissipation given in equation (8–2.19) and the amplification factor in equation (8–2.17),

$$\begin{aligned} |A|^2 &\approx 1 - 2\,|\epsilon|\,c\,(1-\cos\beta) - \epsilon^2[(c+\epsilon)^2 - 1](1-\cos\beta)^2 \\ &\approx [1 - |\epsilon|\,c\,(1-\cos\beta)]^2. \end{aligned} \tag{8–6.2}$$

Here c, the flux-limiting factor, is unity when a first-order method is used and zero when the net diffusion of the algorithm is zero. The amplification factor can also be identified

with the exponential damping rate of the numerically computed solution

$$A \approx e^{-\gamma\ \Delta t} \approx 1 - \gamma\ \Delta t. \tag{8–6.3}$$

From equations (8–5.2) and (8–5.3),

$$\gamma\ \Delta t \approx |\epsilon| c(1 - \cos\beta) \approx \frac{|\epsilon|}{2} c k^2\ \Delta x^2. \tag{8–6.4}$$

The characteristic length scale of the disturbance is $L \equiv 1/k$ and the characteristic damping time is $1/\gamma$. The quantity c is the time- and space-averaged fraction of linear damping used by the composite-flow algorithm, according to equation (8–2.19), to maintain positivity.

Letting Re_{nd} be the effective Reynolds number due to numerical diffusion, we have

$$Re_{nd} \approx \frac{1/\gamma}{L/v} \approx \frac{2L}{c\ \Delta x}\ , \tag{8–6.5}$$

which is independent of the timestep. This equation defines the *cell Reynolds number* for an Eulerian flow. Equation (8–6.5) expresses the highest Reynolds number flow that can be accurately simulated with a grid of the given resolution. It is roughly $2/c$ times the number of computational cells within a structure of characteristic size L.

The highest Reynolds number flow that can be simulated in full detail with linear Eulerian convection, using even spectral methods, has R_e roughly twice the number of cells across the system. This *cell Reynolds number limit* is based on the linear positivity condition, $c = 1$. It does not apply to nonlinear monotone algorithms that vary the flux-limiting factor c locally from cell to cell. Monotone methods require much less overall dissipation and hence are potentially more accurate for simulating high Reynolds number flows than linear convection methods. If the average value of c is $1/100$, which would occur if about one percent of the cells underwent strong flux limiting during each timestep, flows with Reynolds number 2×10^4 could be represented accurately at intermediate and longer wavelengths on a grid with only 100 cells in each direction. In fact, c often decreases during the course of a calculation because short-wavelength structures tend to be smoothed out. A balance is reached between generation and destruction of short wavelengths by physical viscosity and the nonlinear flux-correction process.

When short wavelengths are not dominant, the long wavelengths can be calculated much less dissipatively by using a monotone method than is implied by this cell Reynolds number limit. In these cases, positivity is not a major problem and very small values of c suffice. Even when the short wavelengths have a relatively large amplitude as in turbulent flows, the nonlinear monotone algorithms still propagate the longer wavelengths at a relatively high Reynolds number. Strong smoothing is only applied in a small region, and hence is applied preferentially to short wavelengths. In Chapter 12, this local effect of the nonlinear flux-limiter is interpreted as an effective subgrid turbulence model.

Multiple Reynolds numbers are generally required to describe flow on a computational grid. Specifically, cell Reynolds numbers have been related both to resolution and the numerical damping of long wavelengths. It is useful to think of the long-wavelength Reynolds number as describing the viscous or numerical decay of the large scales, and the short-wavelength Reynolds number as describing the rate of numerical damping required

to prevent round-off error, computational noise, and true turbulent cascade from building up at the grid scale. This short-wavelength numerical smoothing will not affect the long wavelengths appreciably if the phase errors of the intermediate and long wavelengths are small enough. Thus long, slowly decaying wavelengths can be calculated accurately in flows with much higher Reynolds numbers than the usual numerical cell Reynolds number suggests. Short-wavelength structures still exist in the calculation for a finite length of time. They are strongly damped, but they are not dominant and their numerical smoothing, because it is nonlinear, does not noticeably damp the long-wavelength structures.

REFERENCES

Anderson, J.D., Jr. 1995. *Computational fluid dynamics*. New York: McGraw-Hill.

Antiochos, S.K., C.R. DeVore, and J.A. Klimchuk. 1999. A model for solar coronal mass ejections. *The Astrophysical Journal* 510:485–493.

Benedetto, J.J., and M.W. Frazier, eds. 1994. *Wavelets, mathematics and applications*. Boca Raton, Fla.: CRC Press.

Beylkin, G., and J.M. Keiser. 1997. On the adaptive numerical solution of nonlinear partial differential equations in wavelet bases. *Journal of Computational Physics* 132:233–259.

Boris, J.P. 1971. A fluid transport algorithm that works. In *Computing as a language of physics*, 171–189. Vienna: International Atomic Energy Agency.

———. 1976. Flux-corrected transport modules for generalized continuity equations. NRL Memorandum Report, No. 3237. Washington, D.C.: U.S. Naval Research Laboratory.

Boris, J.P., and D.L. Book. 1973. Flux-corrected transport I: SHASTA – A fluid transport algorithm that works. *Journal of Computational Physics* 11:38–69.

———. 1976a. Solution of the continuity equation by the method of flux-corrected transport. *Methods in Computational Physics* 16:85–129.

———. 1976b. Flux-corrected transport III. Minimal-error FCT algorithms. *Journal of Computational Physics* 20:397–431.

Boris, J.P., D.L. Book, and K. Hain. 1975. Flux-corrected transport II: Generalizations of the method. *Journal of Computational Physics* 18:284–283.

Boris, J.P., A.M. Landsberg, E.S. Oran, and J.H. Gardner. 1993. LCPFCT – A flux-corrected transport algorithm for solving generalized continuity equations. NRL Memorandum Report, No. 6410-93-7192. Washington, D.C.: U.S. Naval Research Laboratory. The information in this report, including the LCPFCT software and test problems, is available on the web.

Canuto, C., M.Y. Hussaini, A. Quarternoni, T.A. Zang. 1988. *Spectral methods in fluid dynamics*. Berlin: Springer-Verlag.

Chakravarthy, S.R., and S. Osher. 1985a. A new class of high accuracy TVD schemes for hyperbolic conservation laws. AIAA paper no. 85-0363. Reston, Va.: American Institute of Aeronautics and Astronautics.

———. 1985b. Application of a new class of high accuracy TVD schemes to the Navier-Stokes equations. AIAA paper no. 85-0165. New York: AIAA.

Cheatham, S., E. Oran, and J. Boris. n.d. Asymptotic behavior and Lagrangian invariance of flux-corrected transport. In preparation for *Journal of Computational Physics*.

Colella, P., and P.R. Woodward. 1984. The piecewise parabolic method (PPM) for gas-dynamical simulations. *Journal of Computational Physics* 54:174–201.

Courant, R., K.O. Friedrichs, and H. Lewy. 1928. Über die partiellen differenzengleichungen der mathematischen physik. *Math. Ann.* 100:32–74. See translation in *IBM Journal*, March 1967, 215–234.

Courant, R., E. Isaacson, and M. Reeves. 1952. On the solution of nonlinear hyperbolic differential equations by finite differences. *Communications in Pure and Applied Mathematics* 5:243–255.

Davis, S.F. 1984. *TVD finite difference schemes and artificial viscosity*. ICASE report no. 84-20. Langley Research Center, Hampton, Va.: NASA.

DeVore, C.R. 1989. Flux-corrected transport algorithms for two-dimensional compressible magnetohydrodynamics, NRL memorandum report no. 6544. Washington, D.C.: U.S. Naval Research Laboratory.

———. 1991. Flux-corrected transport for multidimensional compressible magnetohydrodynamics. *Journal of Computational Physics* 92:142–160.

———. 1998. An improved limiter for multidimensional flux-corrected transport. NRL memorandum report. no. 6440–8330. Washington. D.C.: U.S. Naval Research Laboratory.

Dubal, M.R. 1991. Numerical simulations of special relativistic gas flows. *Computer Physics Communications* 64:221–234.

Ellzey, J.L., K.J. Laskey, and E.S. Oran. 1991. A study of confined diffusion flames. *Combustion and Flame* 84:249–264.

Emery, A.F. 1968. An evaluation of several differencing methods for inviscid fluid flow problems. *Journal of Computational Physics* 2:306–331.

Fletcher, C. 1984. The Galerkin method and Burgers' equation. In *Computational techniques for differential equations*, ed. J. Noye, 355–476. New York: North-Holland.

Fletcher, C.A.J. 1988. *Computational techniques for fluid dynamics, I: Fundamental and general techniques* and *II: Specific techniques for difficult flow categories*. New York: Springer.

Giannakouros, J., and G.Em. Karniadakis. 1994. A spectral element-FCT method for the compressible Euler equations. *Journal of Computational Physics* 115:65–85.

Godunov, S.K. 1959. Finite difference methods for numerical computation of discontinuous solutions of the equations of fluid dynamics. *Mat. Sb.* 47:271–306.

Gottlieb, D., M.Y. Hussaini, and S.A. Orszag. 1984. *Theory and applications of spectral methods*, In *Spectral methods for partial differential equations*, eds. R.G. Voigt, D. Gottlieb, and M.Y. Hussaini, 1–55. Philadelphia: SIAM.

Gottlieb, D., and S.A. Orszag. 1977. *Numerical analysis of spectral methods: Theory and applications*. Philadelphia: SIAM.

Gottlieb, D., and E. Turkel. 1976. Dissipative two-four methods for time dependent problems. *Mathematics of Computation* 30:703–723.

Granjouen, N. 1990. The modified equation approach to flux-corrected transport. *Journal of Computational Physics* 91:424–440.

Harten, A. 1974. *The method of artificial compression*. CIMS report, COO-3077-50. New York: Courant Institute, New York University.

———. 1983. High resolution schemes for hyperbolic conservation laws. *Journal of Computational Physics* 49:357–93.

Harten, A., B. Engquist, S. Osher, and S. Chakravarthy. 1987. Uniformly high order accurate essentially nonoscillatory schemes, III. *Journal of Computational Physics* 71:231–303.

Harten, A., and S. Osher. 1987. Uniformly high-order accurate nonoscillatory schemes, I. *SIAM Journal of Numerical Analysis* 24:279–309.

Harten, A., and G. Zwas. 1972. Self-adjusting hybrid schemes for shock computations. *Journal of Computational Physics* 6:568–583.

Hirsch, C. 1988. *Numerical computation of internal and external flows. Vol. 1, Fundamental of numerical discretization*. New York: Wiley.

———. 1990. *Numerical computation of internal and external flows. Vol. 2, Computational methods for inviscid and viscous flows*. New York: Wiley.

Hussaini, M.Y., and T.A. Zang. 1984. In *Spectral methods for partial differential equations*, eds. R.G. Voigt, D. Gottlieb, and M.Y. Hussaini, 119–140. Philadelphia: SIAM.

Huynh, H.T. 1995. A piece-parabolic method for the Euler equations. In *Proceedings of the 12th AIAA CFD Conference*, paper no. 95-1739. Washington, D.C.: American Institute of Aeronautics and Astronautics.

Kaplan, C.R., S.W. Baek, E.S. Oran, and J.L. Ellzey. 1994. Dynamics of a strongly radiating unsteady ethylene jet diffusion flame. *Combustion and Flame* 96:1–21.

Karniadakis, G.Em. 1998. *Spectral/hp element methods for CFD*. Oxford Univ. Press, Oxford: England.

Karpen, J.T., S.K. Antiochos, C.R. DeVore. 1996. Reconnection-driven current filamentation of solar arcades. *The Astrophysical Journal* 460:L73–L76.

Kopal, Z. 1961. *Numerical analysis*, New York: Wiley.

Kreiss, H. and J. Oliger. 1973. *Methods for the approximate solution of time dependent problems*. Global Atmospheric Research Programme (GARP) publication series no. 10. Geneva: World Meteorological Organization and International Council of Scientific Unions.

Kutler, P., and Lomax, H. 1971. The computation of supersonic flow fields about wing-body combination by 'shock capturing' finite difference techniques. In *Proceedings of the Second International Conference on Numerical Methods in Fluid Dynamics*. Vol. 8, *Lecture notes in physics*, ed. M. Holt, 24–29. New York: Springer-Verlag.

Lanczos, C. 1956. *Applied analysis*. Englewood Cliffs, N.J.: Prentice Hall.

Landsberg, A.M. and J.P. Boris. 1997. The virtual cell embedding method: A simple approach for gridding complex geometries. AIAA paper 97-1982, Reston, Va.: American Institute of Aeronautics and Astronautics.

Landsberg, A.M., J.P. Boris, W. Sandberg, and T.R. Young. 1993. Naval ship superstructure design: Complex three-dimensional flows using an efficient parallel method. In *High performance computing 1993: Grand challenges in computing*, ed. A.M. Tentner, 15–20. San Diego: SCS.

Laney, C.B., 1998. *Computational gasdynamics*, Cambridge, England: Cambridge University Press.

Lax, P.D., and B. Wendroff. 1960. Systems of conservation laws. *Communications in Pure and Applied Mathematics* 13:217–237.

——— 1964. Difference schemes for hyperbolic equations with high order of accuracy. *Communications in Pure and Applied Mathematics Math.* 17:381–398.

Lele, S.K. 1992. Compact finite difference schemes with spectral-like resolution. *Journal of Computational Physics* 103:16–42.

LeVeque, R.J. 1992. *Numerical methods for conservation laws*. 2nd ed. Basel: Birkhäuser Verlag.

Löhner, R., and G. Patnaik. 1987. BIC-FEM-FCT: An algorithm for low mach number flows, In *Proceedings, AIAA 8th Computational Fluid Dynamics Conference*. Reston, Va.: AIAA.

Löhner, R., K. Morgan, J. Peraire and M. Vahdati. 1987b. Finite element flux-corrected transport (FEM-FCT) for Euler and Navier-Stokes equations. *International Journal of Numerical Methods in Fluids* 7:1093–1109.

Löhner, R., K. Morgan, M. Vahdati, J.P. Boris, and D.L. Book. 1988. FEM-FCT: Combining unstructured grids with high resolution. *Comm. Appl. Num. Meth.* 4:717–730.

Löhner, R., G. Patnaik, J. Boris, E. Oran, and D. Book. 1987a. Applications of the method of flux corrected transport to generalized meshes. In *Proceedings, Tenth International Conference on Numerical Methods in Fluid Dynamics*, eds. F.G. Zhuang and Y.L. Zhu, New York: Springer-Verlag.

Lomtev, I., C.B. Quillen, and G.Em. Karniadakis. 1998. Spectral/hp methods for viscous compressible flows on unstructured 2D meshes. *Journal of Computational Physics* 144:325–357.

Luo, H., J.D. Baum, and R. Löhner. 1994. Edge-based finite element scheme for the Euler equations. *AIAA Journal* 32:1183–1190.

MacCormack, R.W. 1969. The effect of viscosity in hypervelocity impact cratering. AIAA paper no. 69-352. Reston, Va.: American Institute of Astronautics and Aeronautics.

———. 1971. Numerical solution of the interaction of a shock wave with a laminar boundary layer. In *Proceedings of the Second International Conference on Numerical Methods in Fluid Dynamics*, Vol. 8, *Lecture notes in Physics*, ed. M. Holt, 151–163. New York: Springer-Verlag.

———. 1981. *A numerical method for solving the equations of compressible viscous flow*. American Institute for Aeronautics and Astronautics paper 81-0110. New York: AIAA.

MaMarti, J., and E. Müller. 1996. Extension of the piecewise parabolic method to one-dimensional relativistic hydrodynamics. *Journal of Computational Physics* 123:1–14.

McDonald, B.E., J. Ambrosiano, and S. Zalesak. 1985. The pseudospectral flux correction (PSF) method for scalar hyperbolic problems. In *Proceedings of the Eleventh International Association for Mathematics and Computers in Simulation World Congress*, Vol. 1, ed. R. Vichnevetsky. 67–70. New Brunswick, N.J.: Rutgers University Press.

Moon, B., G. Patnaik, R. Bennett, D. Fyfe, A. Sussman, C. Douglas, and J. Saltz. 1995. Runtime support and dynamic load balancing strategies for structured adaptive applications. In *Proceedings, Seventh SIAM Conference in Parallel Processing*. New York: SIAM.

Morgan, K., and J. Peraire. 1998. Unstructured grid finite-element methods for fluid mechanics. *Reports on Progress in Physics* 61:569–638.

Murawski, K., C.R. DeVore, S. Parhi. and M. Goossens. 1996. Numerical simulations of MHD wave excitation in bounded plasma slabs. *Planetary and Space Sciences* 44:253–265.

von Neumann, J., and R.D. Richtmyer. 1950. A method for the numerical calculation of hydrodynamic shocks. *Journal of Applied Physics* 21:232–257.

Odstrčil, D. 1990. A new optimized FCT algorithm for shock wave problems. *Journal of Computational Physics* 91:71–93.

Oran, E.S., J.W. Weber, Jr., E.I. Stefaniw, M.H. Lefebvre, and J.D. Anderson, Jr. 1998. A numerical study of a two-dimensional H_2-O_2–air detonation using a detailed chemical reaction model. *Combustion and Flame* 113:147–163.

Orszag, S.A. 1972. Comparison of pseudospectral and spectral approximations. *Studies in Applied Mathematics* 51:253–259.

———. 1980. Spectral methods for problems in complex geometries. *Journal of Computational Physics* 37:70–92.

Parhi, S., P. de Bruyne, K. Murawski, M. Goossens, and C.R. DeVore. 1996. Numerical simulations of driven MHD waves in coronal loops. *Solar Physics* 167:181–202.

Patera, A.T. 1984. A spectral element method for fluid dynamics, laminar flow in a channel expansion. *Journal of Computational Physics* 54:468–488.

Patnaik, G., R.H. Guirguis, J.P. Boris, and E.S. Oran. 1987. A barely implicit correction for flux-corrected transport. *Journal of Computational Physics* 71:1–20.

Patnaik, G. and K. Kailasanath. 1996. A new time-dependent, three dimensional, flame model for laminar flames. In *26th Symposium (International) on Combustion*, 899–905. Pittsburgh: The Combustion Institute.

Patnaik, G., K. Laskey, K. Kailasanath, E. Oran, and T. Brun. 1989. FLIC – A detailed two dimensional flame model. NRL memorandum report, no. 6555. Washington, D.C.: U.S. Naval Research Laboratory.

Peyret, R., and T. Taylor. 1982. *Computational methods for fluid flow*. New York: Springer.

Potter, D. 1973. *Computational physics*. New York: Wiley.

Richtmyer, R.D. 1963. *A survey of difference methods for nonsteady fluid dynamics*. NCAR technical note 63-2. Boulder, Colo.: National Center for Atmospheric Research.

Richtmyer, R.D., and K.W. Morton. 1967. *Difference methods for initial-value problems*. New York: Interscience.

Roache, P.J. 1982. *Computational fluid dynamics*. Albuquerque, N. Mex.: Hermosa Publishers.

Rood, R.B. 1987. Numerical advection algorithms and their role in atmospheric transport and chemistry models. *Reviews of Geophysics* 25:71–100.

Strang, F., and G. Fix. 1973. *An analysis of the finite element method*, Englewood Cliffs, N.J.: Prentice Hall.

Suresh, A., and H.T. Huynh. 1997. Accurate monotonicity-preserving schemes with Runge-Kutta time stepping. *Journal of Computational Physics* 136:83–99.

Sweby, P.K. 1984. High resolution schemes using flux limiters for hyperbolic conservation laws. *SIAM Journal of Numerical Analysis* 21:995–1011.

Tannehill, J.C., D.A. Anderson, and R.H. Pletcher. 1997. *Computational fluid mechanics and heat transfer*, Washington, D.C.: Taylor and Francis.

Taylor, T.D., R.B. Myers, and J.H. Albert. 1981. Pseudo-spectral calculations of shock waves, rarefaction waves and contact surfaces. *Comp. Fluids* 4:469–473.

Toro, E.F. 1997. *Riemann solvers and numerical methods for fluid dynamics*, Berlin: Springer-Verlag.

Van Leer, B. 1973. Towards the ultimate conservative difference scheme. I. The quest of monotonicity. In *Lecture notes in physics* 18, eds. H. Cabannes and R. Temam, 163–168. Berlin: Springer-Verlag.

———. 1979. Towards the ultimate conservative difference scheme. V. A second-order sequel to Godunov's method. *Journal of Computational Physics* 32:101–136.

———. 1986. On numerical dispersion by upwind differencing. *Applied Numerical Mathematics* 2:379–384.

Voigt, R.G., D. Gottlieb, and M.Y. Hussaini, eds. 1984. *Spectral methods for partial differential equations*. Philadelphia: SIAM.

Vuillermoz, P., and E.S. Oran. 1992. The effect of energy release on a supersonic reactive mixing layer. In *Proceedings of the 24th Symposium (International) on Combustion*, 395–403. Pittsburgh: The Combustion Institute.

Weber, J.W., Jr., E.S. Oran, J.D. Anderson, Jr., and G. Patnaik. 1997. Load balancing and performance issues for the data parallel simulation of stiff chemical nonequilibrium flows. *AIAA Journal* 35:486–493.

Weber, W.J., J.P. Boris, and J.H. Gardner. 1979. ALFVEN – A two-dimensional code based on SHASTA, solving the radiative, diffusive MHD equations. *Computer Physics Communications* 16:243–265.

Weber, Y.S., E.S. Oran, J.P. Boris, and J.D. Anderson, Jr. 1995. The numerical simulation of shock bifurcation near the end wall in a shock tube. *Physics of Fluids* 7:2475–2488.

Woodward, P., and P. Colella. 1984. The numerical simulation of two-dimensional fluid flow with strong shocks. *Journal of Computational Physics* 54:115–173.

Yee, H.C., R.F. Warming, and A. Harten. 1983. Implicit total variation diminishing (TVD) schemes for steady-state calculations. AIAA paper no. 83-1902. Reston, Va.: American Institute of Aeronautics and Astronautics.

Zalesak, S.T. 1979. Fully multidimensional flux-corrected transport algorithms for fluids. *Journal of Computational Physics* 31:335–362.

———. 1981. Very high order and pseudospectral flux-corrected transport (FCT) algorithms for conservations laws. In *Advances in computer methods for partial differential equations*, Vol. IV, eds. R. Vichnevetsky and R.S. Stepleman, 126–134. New Brunswick, N.J.: International Association for Mathematics and Computers in Simulation (IMACS), Rutgers University.

———. 1984. A physical interpretation of the Richtmyer two-step Lax-Wendroff scheme and its generalization to higher spatial order. In *Advances in Computer methods for partial differential equations*, Vol. V, eds. R. Vichnevetsky and R.S. Stepleman, 491–496. New Brunswick, N.J.: International Association for Mathematics and Computers in Simulation (IMACS), Rutgers University.

———. 1987. Preliminary comparison of modern shock-capturing schemes: Linear advection. In *Advances in computer methods for partial differential equations*, VI, 15–22 eds. R. Vichnevetsky and R.S. Stepelman, 15–22. New Brunswick, N.J.: IMACS, Rutgers University.

———. 1997. Introduction to "flux-corrected transport I: SHASTA – a fluid transport algorithm that works," *Journal of Computational Physics* 135:170–171.

Zhmakin, A.I., and A.A. Fursenko. 1979. *On a class of monotonic shock-capturing difference schemes*. Leningrad: Ioffe Physicotechnical Institute, USSR Academy of Sciences.

Zienkiewicz, O.C. 1977. *The finite element method*. New York: McGraw-Hill.

Zienkiewicz, O.C., and K. Morgan. 1983. *Finite elements and approximation*. New York: John Wiley.

Zienkiewicz, O.C., and R.L. Taylor. (date ?) *The finite elements method, I: Basic formulation and linear problems*. New York: McGraw-Hill.

———. 1991. *The finite elements method, II: Solid and fluid mechanics; dynamics and non-linearity*. New York: McGraw-Hill.

9

Computational Fluid Dynamics: Using More Flow Physics

Chapter 8 described numerical algorithms for solving continuity equations and presented straightforward methods to solve sets of continuity equations to simulate fluid systems. This chapter describes CFD methods that seek to improve the solution by incorporating more of the known flow physics. In some cases, additional constraints are added to the solution of the continuity equations. In other cases, the formulation of the problem itself is changed to use variables other than the primary conserved variables ρ, $\rho\mathbf{v}$, and E. The result is usually a more complicated algorithm and a less general numerical model. Sometimes, however, it can lead to a more accurate solution method for specific classes of problems.

This chapter first considers methods that exploit approximations based on the the flow speed. As discussed in Chapter 2 (Section 2–2.1, Table 2.2), flow speeds are generally divided into five regimes. In order of increasing Mach number, these are: incompressible, subsonic, transonic, supersonic, and hypersonic flows. The boundaries between these regimes loosely mark the appearance or disappearance of different physical phenomena, and each regime has peculiar features that make certain models and solution algorithms more effective. The material presented below describes methods for solving coupled flow problems in three speed regimes which are composites of those listed in Table 2.2.

- *Fast flows* (see Section 9–2). In this regime the fluid velocity is at least a significant fraction of the speed of sound or faster. Compressibility effects, such as shocks, must be resolved.
- *Incompressible flows* (see Section 9–3) are at the opposite limit. Changes in fluid density are negligible in the frame of reference moving with the fluid locally, but large density gradients may be passively convected with the flow. The fluid motion is implicitly coupled throughout the domain by the very fast acoustic waves. The pressure is found from the solution of an elliptic equation that is sensitive to the treatment of boundary conditions.
- *Slow flows* (see Section 9–4) are the most difficult regime to treat. The sound speed is so high that we would like to neglect sound waves, as we do for incompressible flows, but the long-term effects of compressibility cannot be ignored. This happens

when the fluid compresses or expands because of other physical processes such as chemical heat release, radiative cooling, or gravitational effects.

A second way of classifying CFD models is based on the underlying representation of the fluid, as discussed in Chapter 6. Lagrangian representations (see Section 9–5), particle and quasiparticle representations (see Section 9–6), and vortex dynamics representations (see Section 9–7) are distinct representations leading to distinct algorithms for the incompressible, fast, and slow-flow regimes. In this chapter, we discuss the algorithmic aspects and grid definition of variables for these methods.

A third physical variation that can be used to classify flow problems is the degree and manner in which fluid viscosity plays a role. The modifications required to include viscosity, and more generally physical diffusion, are considered briefly as each regime is discussed. The changes in physics these processes lead to are treated extensively in standard fluid dynamics texts.

There is no way that a few pages of text on any of the topics in this chapter can give an in-depth summary of the topic. Instead, the objectives are to provide a useful classification of the methods, to describe their underlying similarities and differences, and to give the reader some pointers to useful references. Again, we highlight those approaches that are most promising and best suited for simulations of reactive flows. Chapters 4, 8, and 9 are the beginning elements for a course in computational fluid dynamics.

9–1. Building Flow Physics into CFD Algorithms

9–1.1. Equations of Compressible and Incompressible Flow

When the pressure gradients in a fluid are large, the accelerations and the velocities also become large, and inertial effects become important. The density of the fluid can vary substantially as strong sound waves and shocks pass. When the pressure gradients and flow velocities are small, even though the pressure itself may be high, the fluid behaves incompressibly. This means that fluid forces created either by inertia or pressure gradients are too small to compress the fluid significantly.

Incompressible Flows

For multidimensional *incompressible* flows with negligible viscosity, we can solve the *primitive equations*,

$$\frac{\partial \rho}{\partial t} + \mathbf{v} \cdot \nabla \rho = 0, \tag{9–1.1a}$$

$$\nabla \cdot \mathbf{v} = 0, \tag{9–1.2}$$

$$\rho\left(\frac{\partial \mathbf{v}}{\partial t} + \mathbf{v} \cdot \nabla \mathbf{v}\right) + \nabla P = \mathbf{f}_e + \nabla \cdot \mu \nabla \mathbf{v}. \tag{9–1.3a}$$

This Eulerian representation uses the *primitive variables* ρ and $\mathbf{v}$ for density and velocity. The term $\mathbf{f}_e$ represents external forces applied to the fluid, for example, gravity, and μ is the viscosity. (See Chapter 2 for variable definitions.) When the fluid is incompressible,

the equation for energy conservation can be replaced by the incompressibility constraint, expressed in equation (9–1.2), which effectively removes sound waves from the system. Equations (9–1.1a) and (9–1.3a), the conservation equations for density and momentum, can also be written in an alternate, Lagrangian form,

$$\frac{d\rho}{dt} = 0, \tag{9–1.1b}$$

$$\rho \frac{d\mathbf{v}}{dt} + \nabla P = \mathbf{f}_e + \nabla \cdot \mu \nabla \mathbf{v}. \tag{9–1.3b}$$

Equation (9–1.1b) states that the density is a conserved quantity, moving with the flow. The density remains constant in the fluid frame of reference, even though there may be large, moving density variations in the laboratory frame. The term $\mathbf{f}_e$ is ignored for most of the discussions in this chapter.

Alternately, equations (9–1.2) and (9–1.3), which govern the evolution of the fluid velocity, may be replaced by a mathematically equivalent system, the *vorticity dynamics* equations,

$$\boldsymbol{\omega} \equiv \nabla \times \mathbf{v}, \tag{9–1.4}$$

$$\mathbf{v} = \nabla \times \boldsymbol{\psi}, \tag{9–1.5}$$

$$\nabla^2 \boldsymbol{\psi} = \boldsymbol{\omega}, \tag{9–1.6}$$

$$\frac{\partial \boldsymbol{\omega}}{\partial t} + \nabla \cdot (\mathbf{v}\boldsymbol{\omega}) = -\nabla \times \left(\frac{\nabla P}{\rho} \right) + \nabla \times \left(\frac{\nabla \cdot \mu \nabla \mathbf{v}}{\rho} \right). \tag{9–1.7}$$

Here $\boldsymbol{\psi}$ is the vector stream function and $\boldsymbol{\omega}$ is the vector vorticity. In two dimensions, both the stream function and vorticity are scalars. Equations (9–1.5) through (9–1.7) become

$$\mathbf{v} = \widehat{\mathbf{z}} \times \nabla \psi, \tag{9–1.8}$$

$$\nabla^2 \psi = \omega, \tag{9–1.9}$$

$$\frac{\partial \omega}{\partial t} + \nabla \cdot (\mathbf{v}\omega) = -\nabla \times \left(\frac{\nabla P}{\rho} \right), \tag{9–1.10}$$

where $\widehat{\mathbf{z}}$ is the unit vector in the z-direction. In either formulation, the pressure is found from an elliptic equation,

$$\nabla^2 P = \frac{1}{\rho} \nabla \rho \cdot \nabla P - \rho \nabla \mathbf{v} : \nabla \mathbf{v}, \tag{9–1.11}$$

obtained by taking the divergence of equation (9–1.3) and using equation (9–1.2). The density is advanced using equation (9–1.1a) or equation (9–1.1b). For incompressible two-dimensional flows at constant density,

$$\frac{d\boldsymbol{\omega}}{dt} \equiv \frac{\partial \boldsymbol{\omega}}{\partial t} + \mathbf{v} \cdot \nabla \boldsymbol{\omega} = 0. \tag{9–1.12}$$

Equation (9–1.12) shows that, when the viscosity is zero, the two-dimensional vorticity is also a conserved quantity moving with the fluid.

Both the primitive equation formulation and the vorticity dynamics formulation involve convective equations such as equations (9–1.1), (9–1.3), (9–1.7), and (9–1.10). In addition, both formulations involve Poisson-like equations, such as equation (9–1.6), (9–1.9), and (9–1.11), whose numerical solution procedures are discussed in Chapter 10.

Compressible Flows

For compressible flows, we replace equation (9–1.1a) by the full continuity equation,

$$\frac{\partial \rho}{\partial t} + \mathbf{v} \cdot \nabla \rho + \rho \nabla \cdot \mathbf{v} = 0, \tag{9–1.13}$$

and equation (9–1.2) either by an isentropic pressure equation,

$$\frac{\partial P}{\partial t} + \mathbf{v} \cdot \nabla P + \gamma P \nabla \cdot \mathbf{v} = 0, \tag{9–1.14}$$

or by a conservation equation for the energy density E,

$$\frac{\partial E}{\partial t} + \nabla \cdot (\mathbf{v}(P + E)) = 0. \tag{9–1.15}$$

For an ideal gas,

$$E = \epsilon + \frac{1}{2}\rho v^2. \tag{9–1.16}$$

When $\nabla \cdot \mathbf{v} \neq 0$, the fluid velocity $\mathbf{v}$ is the sum of terms derived from a vector potential and a scalar potential,

$$\mathbf{v} = \nabla \times \boldsymbol{\psi} + \nabla \phi. \tag{9–1.17}$$

This is the same as equation (2–2.14) discussed in Chapter 2, except that here we have omitted the particular solution ϕ_p. Both $\boldsymbol{\psi}$ and ϕ satisfy Poisson equations. Reconstructing the velocity field from equation (9–1.17) by solving these Poisson equations, given the velocity and divergence source terms, is called the *div-curl problem*. The source in the Poisson equation for ϕ may be written

$$\sigma \equiv \nabla \cdot \mathbf{v}, \tag{9–1.18}$$

where the divergence σ satisfies the convective equation

$$\frac{\partial \sigma}{\partial t} + \nabla \cdot (\mathbf{v}\sigma) = \sigma^2 - \nabla \mathbf{v} : \nabla \mathbf{v} + \frac{\nabla \rho \nabla P - \rho \nabla^2 P}{\rho^2}, \tag{9–1.19}$$

which results from taking the divergence of equation (9–1.3). For compressible flow, there is generally no advantage to using the $\boldsymbol{\omega} - \sigma$ formulation.

Both equations (9–1.14) and (9–1.15) are convection equations and can be solved by the same methods, but there is an important physical distinction between them. Equation (9–1.14) can only be used to describe flows without shocks, that is, isentropic flows, since entropy is not conserved in shocks. Using the energy conservation equation (9–1.15)

introduces an added difficulty, the need to extract the pressure P from equation (9–1.16). When the fluid kinetic energy dominates the internal energy, that is, when $P \ll E$, negative pressures can result from small truncation errors in either the total energy density or the kinetic energy density. These derived negative pressures can be made positive and monotone by using the isentropic pressure equation to provide a lower bound on the pressure, but this requires solving an additional continuity equation.

9–1.2. Characteristic Trajectories

Here we introduce the concepts of flow characteristics, which appear in this chapter and then again in Section 10–1 on boundary conditions. The development of the method of characteristics was described in Courant and Friedrichs (1948) and Courant and Hilbert (1962), and is discussed in all texts dealing with compressible flow (see, for example, Liepmann and Roshko [1957] and Tannehill, Anderson, and Pletcher [1997]).

Consider an ideal, one-dimensional, planar, constant-entropy compressible flow, with fluid velocity v and sound speed c_s defined by

$$c_s = \sqrt{\left.\frac{\partial P}{\partial \rho}\right|_s}, \tag{9–1.20}$$

where the derivative is taken at constant entropy. Information from any point in the flow propagates according to the equations

$$\frac{dx}{dt} = v \tag{9–1.21}$$

and

$$\frac{dx}{dt} = v \pm c_s. \tag{9–1.22}$$

Equation (9–1.21) defines a trajectory C_o, along which the entropy s is convected without change. In addition, when s is constant throughout the flow, the *Riemann invariants*,

$$R_\pm = v \pm \int \frac{dP}{\rho c_s}, \tag{9–1.23}$$

are constant along the trajectories $C_\pm$ defined by equation (2–2.27) (Landau and Lifshitz 1959; Courant and Friedrichs 1948). The $C_\pm$ trajectories follow sound waves moving forward and backward relative to the flow. The C_o trajectory follows a particle path. Variations in the entropy are convected according to the adiabatic equation

$$\frac{\partial s}{\partial t} + v\frac{\partial s}{\partial x} = 0. \tag{9–1.24}$$

Equations (9–1.20) through (9–1.24) completely describe flows in which the entropy is constant following the fluid motion. The quantities C_o and $C_\pm$ are called the *characteristic trajectories*, or *characteristics* of the flow.

In flows for which the entropy is not constant, two types of discontinuities, or interfaces between different states of the same medium, can occur. One type is the *contact discontinuity*, across which density, energy, tangential velocity, and entropy may be discontinuous, but pressure and normal velocity are continuous. The other type is the *shock*, across which all dependent variables change discontinuously. Contact discontinuities propagate at the fluid velocity, $\mathbf{v}$, that is, along C_o. Shock waves move with a velocity v_s which uniquely determines the discontinuous changes in ρ, $\mathbf{v}$, and P. In the limit of very weak shocks, the shock trajectory reduces to an acoustic disturbance, one of the two trajectories $C_\pm$.

9–1.3. Time Integration

A fundamental decision in using finite-difference approaches is what method to use for the time integration. This decision is dominated by the question of whether to choose an explicit or an implicit method. Another important consideration is how much added computer memory is required to store previous values of the relevant variables. A number of implicit formulations have been described throughout this book, starting in Chapter 4. The generic form for the evolution of a variable ρ may be written as

$$\rho^{n+1} = (1 - \theta)F_e(\rho^n) + \theta F_i(\rho^{n+1}), \tag{9–1.25}$$

where ρ is to be evaluated at the new time, t^{n+1}, in terms of values at the old time, t^n, and where F_e and F_i are some functions of ρ. The quantity θ is the implicitness parameter: $\theta = 0$ for an explicit solution, $\theta = 1$ for an implicit solution, and $\theta = \frac{1}{2}$ for a time-centered, semi-implicit solution.

Whenever $\theta \neq 0$, the method usually requires solving a large, but sparse, matrix equation at each timestep. The best cases may only require inverting a tridiagonal matrix system of equations. Though the cost per timestep can be quite high compared with explicit methods, the advantage of implicit methods is their ability to remain numerically stable with longer timesteps. There are two questions to consider:

1. What are the important time scales we need to resolve?
2. What is the relation of the timestep we would like to take to the stability limits of the algorithm?

Whether we want to use an implicit or an explicit time integration depends on the answers to these questions.

Suppose we wish to resolve a particular flow structure in a convection problem, and this structure varies appreciably in time τ. No calculation with $\Delta t \gg \tau$ can accurately describe the details of this structure. This is analogous to the case of the finite spatial mesh. If the mesh is too coarse, the structure cannot be resolved. If the physical time scale τ is less than the timestep required by the Courant stability condition, an explicit method is sufficient. If τ is appreciably larger than the timestep required to satisfy the Courant condition, that is, if the structure varies very slowly, it is worthwhile to consider an implicit method while still ensuring $\Delta t < \tau$.

Consider a situation in which the phenomenon to be resolved is not the fastest effect in the problem. An example of this is the expanding flow that results as air is blown out of

a whistle. It would be natural to consider using an implicit method because the speed of sound is about two orders of magnitude larger than the flow speed. We do not really need to resolve sound waves as long as ignoring them does not seriously effect the accuracy of the calculation of the slower flow.

Unfortunately, this last sentence contains the crux of a serious physical and numerical problem. A major difficulty with implicit convection algorithms is that they tend to damp high-frequency modes. This means that acoustic energy may be incorrectly deposited as heat near the whistle. A time-centered method, or one that is nearly centered, allows much of the acoustic energy to propagate away as sound without heating the air. Again, the choice of method depends strongly on the specific problem.

9–1.4. Calculating Fast and Slow Flows

Modeling energetic reacting flows and combustion systems requires the description of flows that are subsonic, such as flames, and flows that are supersonic, such as shocks and detonations. In the best of all possible worlds, all regimes of flow could be treated with one, all-inclusive numerical algorithm. A great deal of effort has been invested in developing such algorithms, which invariably use implicit methods. Nonetheless, there is a high price to pay for being able to use a single algorithm for both fast and slow flows. This price translates into many computer operations per timestep spent in solving expensive elliptic or matrix equations that may lead to physically questionable effects.

There are several fundamental conflicts inherent in constructing truly general CFD methods. To be economical for slow flows, the method should handle sound waves implicitly. This is generally done by solving the pressure equation instead of the energy equation and often requires artificial viscosity to be stable. For the benefit of being able to take long timesteps when the flow velocities are low, the spatial resolution is poor and shock fronts are unnecessarily thick when the flows are fast.

Furthermore, there is danger in using an implicit algorithm for supersonic flows, because these algorithms transmit numerical precursors ahead of shocks at essentially infinite speed. Many explicit methods for supersonic and hypersonic flows use numerical algorithms that do not perform well when used for subsonic flows. As discussed in Chapter 12 on turbulence, this is the case if the underlying algorithm is low-order or nonmonotone. The result is that excess numerical damping, used to stabilize the nonlinear effects in supersonic flows, appears as rapid damping when the flow velocity is small compared to the sound speed.

We generally recommend different methods for very slow and very fast flows. Such methods are somewhat limited in generality, but offer advantages in accuracy, speed, and simplicity compared to global or composite implicit algorithms. For this reason, this chapter discusses methods for the different flow regimes separately.

9–2. Methods for Fast Flows

Fast flows have characteristic velocities ranging from below the sound speed to far above it. This includes flows that are subsonic, transonic, supersonic, and hypersonic. While there are usually no shocks in subsonic flows, compressibility effects can still be very

important. This is a borderline regime. For some problems, subsonic flows should be treated with the implicit or slow-flow methods described in Sections 9–3 and 9–4. Other subsonic problems must be treated by solutions of the coupled continuity equations that fully resolve the sound waves. One example of this type of problem is acoustic-vortex interactions in combustion chambers where the vortices are moving slowly compared to the sound speed.

In the transonic regime, the finite sound speed and compressional effects become important and play a major role as the flow speed approaches the sound speed. The geometry becomes crucial near Mach one, where the development of shocks depends on subtle variations in geometry and the structure of the boundary layers. Transonic problems are of particular importance to aerodynamics.

In the supersonic flow regime, the shocks must be well localized and the nearly discontinuous jump in the fluid variables across shocks must be calculated accurately. Here the trade-off is between work invested in special purpose programming to resolve shocks and contact discontinuities and work spent on more finely resolved calculations that do not resolve shocks as well. In the last ten years, a point of diminishing returns has been reached for supersonic-flow algorithms. The most direct way to get significantly better calculations is not through efforts to improve the algorithms, but through applications of adaptive gridding, as discussed in Chapter 6.

Because the fluid velocity in the hypersonic flow regime can be much larger than the sound speed, it may be difficult to calculate the temperature accurately and to conserve total energy locally. Calculating the thermal energy, in a formalism that strictly conserves energy, requires taking the difference of the total energy density and the kinetic energy density in regimes where they are nearly equal. The result is the possibility of large errors from taking differences of large numbers. These errors are compounded by the high temperatures accompanying hypersonic flows. Atmospheric reentry problems, for example, fall in this hypersonic regime.

In all supersonic flows where time variation is important, the CFL condition (Chapter 8) restricts the computational timestep and the grid spacing according to

$$\left.\frac{c_s}{\Delta x}\right|_{\max} < \frac{1}{\Delta t}, \tag{9–2.1}$$

where c_s is the sound speed. This condition on the sound waves is similar to limiting the flow of material through the grid to one cell per timestep. Typically a timestep is found by using the sum $(c_s + |v|)$ instead of just c_s, so that the required timestep becomes even smaller. Where there is a strong shock, when sound speeds are generally comparable to the flow speeds, there is no obvious advantage to using an implicit method. Thus explicit, Eulerian methods are most effective. An implicit method could be used to avoid an even smaller timestep restriction imposed, for example, by a very fine grid needed to resolve a boundary layer elsewhere in the calculation.

In many cases, whether to adopt an Eulerian or a Lagrangian formulation in the supersonic regime depends on the duration of the transient problem. In many explosion, detonation, and shock calculations, the calculation is over by the time the sound waves or shocks have crossed the system once or twice. Then a Lagrangian grid will not become excessively distorted and may be well suited to tracking material interfaces in the flow. To

date, however, most aerodynamics calculations have "marched" or "relaxed" to a steady-state solution of the Navier-Stokes equations on fixed or adaptive Eulerian grids. Further, as shown in Section 9–5, no finite-pressure flow is entirely Lagrangian. For these reasons, we emphasize Eulerian methods in the remainder of this section.

Shock-Tracking and Shock-Capturing Methods

One way to classify different numerical approaches for treating shock propagation in otherwise smoothly varying fluids is by whether the method is a *shock-tracking method* or a *shock-capturing method*. These are fundamentally different approaches. In shock tracking, a shock is considered a discontinuity whose position is determined from analytic properties of the solution. In shock capturing, the position of the shock is determined from a numerical solution of the Euler equations.

Shock-tracking methods were originally called *characteristic methods* and now are sometimes called *shock-fitting methods*. These approaches rely on a representation that merges a description of the gas between the shocks with a direct Lagrangian representation of the shock surface itself. The techniques of *interface tracking*, discussed in Chapter 10, are applied to shock and contact-surface discontinuities. Flow characteristic and Riemann solutions are important in determining where the shock moves in a timestep.

The second class of approaches are called shock-capturing methods. These methods attempt to represent the movement of a shock, which is physically much thinner than one computational cell, by "capturing" it on the usual finite-volume or finite-element grid representation. Fluid variables are treated as if they were everywhere continuous, differentiable functions. For example, the actual thickness of a shock is typically on the order of 50 to 100 Å in a gas, but the computational cells are typically centimeters or even meters in size. Because of this fundamental disparity in scales, the algorithms propagating these shocks, such as the various monotone methods of Chapter 8, must be carefully constructed. Extensions of these ideas tailored specifically to fluid dynamics shocks are considered in the remainder of Section 9–2. Such shock-capturing approaches are prevalent today because they are much better suited than characteristic methods for solving complex fluid and reactive-flow problems. In such problems, there are dynamic phenomena almost everywhere in the computational domain that need to be resolved, not just at isolated fronts.

Central and Upwind Differencing

There are a number of ways to classify the numerical methods for shock-capturing. One of these differentiates by the types of differences that are used, whether they are centered or upwinded. Standard forms of FCT, for example, use central-difference methods, although the FCT techniques could be applied to upwind schemes as well. There are problems, however, where upwinding can be useful, such as following and capturing shocks, and some for which it may be a detriment, such as resolving the detailed behavior of sound waves in a compressible fluid.

Upwind-differencing methods are *one-sided* algorithms that attempt to discretize the equations using finite differences that are biased in the direction of the fluid flow. The concept goes back to the first explicit upwind methods by Courant, Isaacson, and Reeves (1952). An example used already in this book is the one-sided, upwinded, donor-cell algorithm

introduced in Section 4–4 to solve the square-wave convection problem. For these methods, two models have been proposed to determine the interaction between computational cells. One is alternately called the *Godunov approach* or *Riemann approach*, which gives rise to numerical methods called *flux-difference splitting*. Another is called the *Boltzmann approach*, which gives rise to numerical methods called *flux-vector splitting*. An excellent review that summarizes the essence of both approaches is given by Harten, Lax, and Van Leer (1983), and there are also basic descriptions in Van Leer (1982), Harten (1983), Roe (1986), and in textbooks such as those by Hirsch (1990) or Toro (1997).

9–2.1. The Riemann Problem and Godunov Methods

The solution to the one-dimensional Riemann problem describes the evolution of a single planar discontinuity separating two different but uniform fluid regions. For one-dimensional Euler equations, solutions may consist of shocks, contact discontinuities, and rarefactions. The discontinuity between the two regions might be in pressure or density, a situation that occurs in the standard shock-diaphragm problem, and it could also be a material interface, a contact surface, or a rarefaction wave. At shocks and contact discontinuities, conservation laws give analytic jump conditions that can be used to determine how the discontinuity evolves in time. Even with ideal equations of state, the resulting equations are nonlinear and some iteration may be needed to find the solutions. Good descriptions of the Riemann problem and its solutions are given in all texts on shocks and compressible flows (see, for example, Liepmann and Roshko [1957], Toro [1997]).

Godunov first incorporated solutions of the Riemann problem into a finite-difference method to improve accuracy at a discontinuity (Godunov 1959; see also Richtmyer and Morton 1967). The fundamental and novel idea of the method involves two steps. The first step solves a Riemann problem cell by cell or region by region in the flow. The second step pieces these local analytic solutions together on the computational grid. The Riemann problem is solved for the discontinuities at each cell interface to determine where the shocks, contact discontinuities, and rarefaction fans have moved after a time Δt. Figure 9.1 shows two cell interfaces along the horizontal axis with time on the vertical axis. At time t, there are discontinuities at the cell interfaces that later propagate outward as a function of time. It is important in this method that waves originating at the interface at the beginning of the timestep do not reach the cell boundary at the end of a timestep. This is assured by keeping Δt small enough, that is, by using essentially the same timestep condition as in an explicit finite-difference method. The result is that conditions throughout each cell are known "exactly." This process produces provisional values of v and P, which are used in the subsequent step.

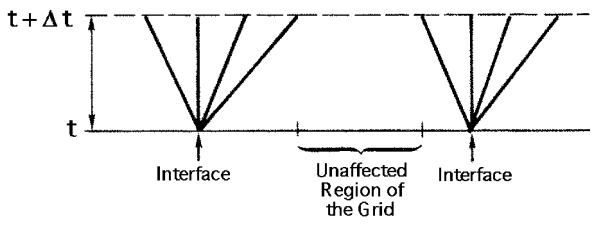

Figure 9.1. Solutions of the Riemann problem at cell interfaces.

A basic assumption in the original formulation is that the solutions are initially piecewise constant in each computational cell. This means that the flow variables have step discontinuities in passing from one cell to the next. The solution procedure then uses the provisional values to piece together the discontinuous solutions. This is done using a first-order interpolation method, so that the method is monotone but diffusive everywhere except at the discontinuity. (See Chapter 8 for a discussion of monotonicity.) The discontinuous solutions do a reasonable job of approximating a solution with discontinuities, so that shocks should be well represented. Because transported fluxes are computed for all three continuity equations together, there does not need to be any unphysical decreases in entropy. It is interesting that some measure of upwinding is enforced just through using the Riemann solutions. This method was initially designed to solve shock problems, and produced very encouraging results. The major drawback is the intrinsic first-order smoothing of the underlying algorithm in smooth regions.

Since Godunov's paper, there have been major efforts to extend the approach. One direction taken is to find the solutions for various different Riemann problems in forms suitable for use in algorithms solving the Euler equations. For example, challenging aspects of generalizing the Riemann problem are related to the orientation of the discontinuities relative to the grid in multidimensions, the extension to geometries other than Cartesian, and the inclusion of more complex physics, such as real equations of state and magnetohydrodynamics. In these situations, an analytic Riemann solution for assigning and following the system characteristics does not exist, so further approximations must be made. For example, one way to convert the multidimensional problem formulation back to locally solvable Riemann problems is to use timestep splitting. Then

$$\frac{df}{dt} = \frac{df_x^R}{dt} + \frac{df_y^R}{dt} + g(f), \tag{9–2.2}$$

where the terms df_x^R/dt and df_y^R/dt represent terms contributing to separable and solvable one-dimensional Riemann problems in the x- and y-directions, respectively. The remaining term, $g(f)$, must contain all of the effects not representable as local Riemann problems. This includes geometric effects that occur even in one-dimensional cylindrical and spherical coordinate systems.

It is natural but misleading to pay careful attention to the parts of a problem that we can solve analytically, the df_x^R/dt and df_y^R/dt terms, for example, while paying little attention to the correction terms, $g(f)$, which can be just as important to the overall solution. Many methods, that are excellent for specific one-dimensional Riemann test problems, cannot be applied with confidence to more complicated flows.

Another concern for chemically reacting flows is energy conservation. Some techniques based on the Riemann solution have trouble guaranteeing the exact conservation of global mass, momentum, or energy. This is true of Glimm's random choice method (Glimm 1965; Chorin 1976, 1977) and algorithms that advance derived quantities such as temperature and entropy, rather than the conserved variables ρ, $\rho\mathbf{v}$, and E.

Extensive work has been directed at finding approximate solutions to the Riemann problems even for nonideal fluids. Review of the development of Riemann solvers in the context of convection problems were given by Van Leer (1979) and Roe (1981), and is

now incorporated in many textbooks dealing with numerical methods for compressible flow, such as Hirsch (1990) and Toro (1997). We note that Roe (1985) incorporates an approximate Riemann solver that is less expensive without obvious deleterious effects. Significant efforts have gone into transforming to local coordinate systems where the Riemann problem is valid (Eidelman 1986) but computational cost and global conservation remain issues.

Another approach to improving algorithms for Godunov methods is to continue to ensure monotonicity while increasing the order of the underlying convection algorithm. As described in Chapter 8, this requires either incorporating a flux limiter or adding artificial diffusion in the stage where the provisional variables are interpolated back onto the grid. This approach has produced a number of higher-order monotone methods that maintain sharp discontinuities and have more accurate representations of convection. Higher-order methods were developed by Van Leer (1973, 1979) and incorporated in MUSCL which uses a Riemann solver. For linear advection, such as in the square-wave test problem, MUSCL reduces to a linear hybridization scheme giving results very similar to second-order FCT. The *piecewise-parabolic method* (*PPM*) (Colella and Woodward 1984; Woodward and Colella 1984) combines a Riemann solver and a high-order convection algorithm. Here the function in each computational cell is assumed to be piecewise parabolic, instead of piecewise constant, and the effect is that the solution is more accurate. Roe (1981, 1985) developed SUPERBEE, a high-order method using a Riemann solver. More recent Godunov methods are based on second- or even higher-order convection algorithms. The combination of high-order monotone convection and approximate Riemann solvers has led to a series of robust methods for problems with shocks.

9–2.2. The Boltzmann Gas-Kinetic and Flux-Vector Splitting Methods

Flux-vector splitting is based on the early work by Sanders and Prendergast (1974), Steger and Warming (1981), and later of Van Leer (1982). As with the Godunov methods, this approach attempts to build information about the directional properties of the flow of information into the algorithm. No solution of a Riemann problem is involved. The directional information is found by taking advantage of a mathematical feature of the Euler equations, a linear algebra concept called the *homogeneity property* (Toro 1997). In fact, the starting point for flux-vector splitting is best described using relatively straightforward linear algebra.

Consider the one-dimensional Euler equations written as

$$\frac{\partial \mathbf{U}}{\partial t} + \frac{\partial \mathbf{F}(\mathbf{U})}{\partial x} = 0. \tag{9–2.3}$$

Here

$$\mathbf{U} = \begin{pmatrix} U_1 \\ U_2 \\ U_3 \end{pmatrix} = \begin{pmatrix} \rho \\ \rho u \\ E \end{pmatrix}, \quad \text{and} \quad \mathbf{F} = \begin{pmatrix} F_1 \\ F_2 \\ F_3 \end{pmatrix} = \begin{pmatrix} \rho u \\ \rho u^2 + P \\ u(E + P) \end{pmatrix}. \tag{9–2.4}$$

The Euler equations are in *conservative form*, meaning that $\mathbf{U}$ is a vector of the conserved variables, and $\mathbf{F}$ is a vector of fluxes.

Equation (9–2.3) can be written as

$$\frac{\partial \mathbf{U}}{\partial t} + \mathbf{A}(\mathbf{U})\frac{\partial \mathbf{U}}{\partial x} = 0 \tag{9–2.5}$$

where $\mathbf{A}$ is the Jacobian matrix

$$\mathbf{A}(\mathbf{U}) = \frac{\partial \mathbf{F}}{\partial \mathbf{U}} = \begin{pmatrix} \dfrac{\partial F_1}{\partial U_1} & \dfrac{\partial F_1}{\partial U_2} & \dfrac{\partial F_1}{\partial U_3} \\ \dfrac{\partial F_2}{\partial U_1} & \dfrac{\partial F_2}{\partial U_2} & \dfrac{\partial F_2}{\partial U_3} \\ \dfrac{\partial F_3}{\partial U_1} & \dfrac{\partial F_3}{\partial U_2} & \dfrac{\partial F_3}{\partial U_3} \end{pmatrix}$$

$$= \begin{pmatrix} 0 & 1 & 0 \\ \frac{1}{2}(\gamma-3)u^2 & (3-\gamma)u & (\gamma-1) \\ \left(\frac{1}{2}(\gamma-2)u^3 - \dfrac{c_s^2 u}{(\gamma-1)}\right) & \left(\dfrac{3-2\gamma}{2}u^2 + \dfrac{c_s^2}{\gamma-1}\right) & \gamma u \end{pmatrix} \tag{9–2.6}$$

This has been evaluated directly from $\mathbf{F}$ and $\mathbf{U}$ in equations (9–2.4), assuming also that the total energy $E = \frac{1}{2}\rho(u^2 + \epsilon)$. For an ideal-gas equation of state, $\epsilon = \rho\,(P/(\gamma-1))$ and $\gamma = c_P/C_V$, and the speed of sound is $c_s = \sqrt{(\gamma P/\rho)}$.

Here the homogeneity property means that

$$\mathbf{F}(\mathbf{U}) = \mathbf{A}(\mathbf{U})\,\mathbf{U} \tag{9–2.7}$$

is satisfied by the Euler equations, as can be shown by multiplying $\mathbf{A}(\mathbf{U})$ by $\mathbf{U}$ and so reproducing $\mathbf{F}(\mathbf{U})$ directly. This forms the basis for flux-vector splitting. We diagonalize $\mathbf{A}$ by finding the matrix of eigenvectors $\mathbf{T}$ such that

$$\mathbf{T}^{-1}\mathbf{A}\,\mathbf{T} = \Lambda = \begin{pmatrix} \lambda_1 & 0 & 0 \\ 0 & \lambda_2 & 0 \\ 0 & 0 & \lambda_3 \end{pmatrix} = \begin{pmatrix} u & 0 & 0 \\ 0 & u+c_s & 0 \\ 0 & 0 & u-c_s \end{pmatrix}, \tag{9–2.8}$$

where Λ is a diagonal matrix of eigenvalues of $\mathbf{A}$. This matrix has two positive eigenvalues and one negative eigenvalue. We break this matrix into two parts and write it as

$$\Lambda = \Lambda^+ + \Lambda^- = \begin{pmatrix} u & 0 & 0 \\ 0 & u+c_s & 0 \\ 0 & 0 & 0 \end{pmatrix} + \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & u-c_s \end{pmatrix} \tag{9–2.9}$$

and

$$\mathbf{F} = \mathbf{F}^+ + \mathbf{F}^- = \mathbf{A}^+\,\mathbf{U} + \mathbf{A}^-\,\mathbf{U}, \tag{9–2.10}$$

where

$$\mathbf{A}^+ = \mathbf{T}\,\Lambda^+\,\mathbf{T}^{-1} \quad \text{and} \quad \mathbf{A}^- = \mathbf{T}\,\Lambda^-\,\mathbf{T}^{-1}. \tag{9–2.11}$$

Finally, we write

$$\frac{\partial \mathbf{U}}{\partial t} + \frac{\partial \mathbf{F}^+}{\partial x} + \frac{\partial \mathbf{F}^-}{\partial x} = 0. \tag{9–2.12}$$

Equation (9–2.12) is the starting place for flux-vector splitting. Now the approach is to replace the F^+ derivative with a backward difference, since F^+ is associated only with information coming from upstream, and replace the F^- derivative with a forward difference, since F^- is associated only with information coming from downstream. There are many ways to specify these derivatives, some of which make the solution monotone, high-order, or even implicit. There have also been efforts to make this approach multidimensional. Flux-vector splitting is now described in great detail in a number of textbooks (see, for example, Toro [1997], Hirsch [1990], and Tannehill et al. [1997]).

The Boltzmann gas kinetic (BGK) method is a relatively new and promising CFD approach that also makes use of directional information. Flux-vector splitting solves the gas-dynamic equations by computing the flux from one cell to the next as the sum of fluxes, each being upwinded by following the split flux vectors along the appropriate characteristic. The BGK approach (Xu et al. 1996; Kim et al. 1997a,b) also considers physical information entering each cell from both directions to construct the effective fluxes. In this case, however, the specifics of the collisional particle distribution function are approximated during relaxation to the local Maxwellian. These methods solve self-consistently for the left- and right-moving fluxes of particles carrying the corresponding amounts of mass, momentum, and energy from one cell to the next. In the references cited, the reconstruction of the solution profile follows the Van Leer MUSCL approach, while the BGK analysis is used to compute the fluxes on both structured and unstructured adaptive meshes.

Algorithms for fluid solutions based on the essentially collisionless Boltzmann equation began with the "Beam scheme" (Sanders and Prendergast 1974) and have since been extended and modified by a number of authors. More recent work has centered on extending the idea to collisional fluids to reduce the viscosity implicit in free-streaming, nearly collisionless solutions. An introduction with references is given in Xu et al. (1996). Current versions of the method give monotone solutions of complex shock problems quite as good as other monotone methods based on Riemann solvers and flux-limiting algorithms.

9–2.3. Comparisons and Caveats

As shown in Chapter 8, methods for solving a single continuity equation are demonstrated and tested by applying them to linear convection of a given profile, such as a square wave or Gaussian. Methods for solving Euler equations are usually compared by how well they reproduce the analytic solution of a one-dimensional propagating shock or an exploding-diaphragm problem. Figure 9.2 shows the "Sod problem," which is a fairly standard test that came to note after the review by Sod (1978). Other typical problems include a temperature jump instead of a pressure jump in the initial conditions, and a shock interacting with a contact discontinuity. The initial conditions for the exploding-diaphragm test problem are shown in Figure 9.2a, and the general form of

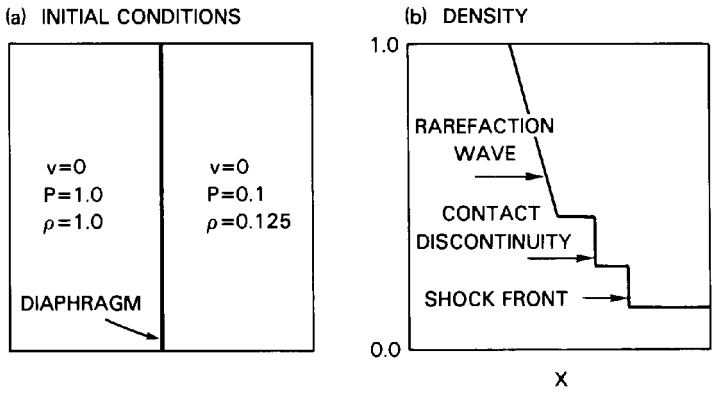

Figure 9.2. Sod's Riemann test problem. (a) Initial conditions. (b) Solution after some time.

the solution after a short time is shown in Figure 9.2b. The calculations presented below used 100 evenly spaced cells and a fixed timestep chosen such that the Courant number $|c_s + v|\Delta t/\Delta x = 0.5$.

Before we describe the results, a caveat concerning the evaluation of algorithms is in order. A method that works "perfectly" for all problems does not exist. *Different methods look better or worse, depending on the specific test problem.* It is common in journal articles describing a new method (not so much in textbooks) for algorithms to be presented in ways that show their best features. For example, a method that uses an exact Riemann solver should do a perfect job when solving an idealized shock-tube problem. This does not necessarily mean that this method would be the most accurate, robust, efficient, or cost-effective method in a complex reactive-flow problem. Therefore, we warn readers to reserve judgment about conclusions presented from one or even several test problems.

Figures 9.3 through 9.6 show the results of applying several rather different types of methods to this exploding-diaphragm test problem. These are:

a. Lax-Wendroff, Figure 9.3. The algorithm is described in Section 8–4.1, and is used as described there, with no artificial viscosity. The application of this method to linear convection was shown in Figure 4.6 and corresponds to a standard version of the MacCormack algorithm.
b. LCPFCT, Figure 9.4. This is the nonlinear monotone flux-corrected transport algorithm described in Section 8–3. It is fourth order in phase, and second order in time. Application of this to linear convection was shown in Figure 4.6 and Figure 8.4. No auxiliary flux synchronization was attempted.
c. The Godunov method, Figure 9.5. This is the linear first-order monotone method originally proposed by Godunov.
d. SUPERBEE, in Figure 9.6. This is the algorithm developed by Roe (1985) that includes an approximate Riemann solver.

Each of Figures 9.3 through 9.6 shows pressure, density, velocity, and temperature profiles at sixty three steps after the diaphragm has burst. During this time, the leading shock has moved twenty five cells downstream. The theoretical solution to this Riemann problem is drawn as a solid line and the calculations are shown as dots. In the test shown,

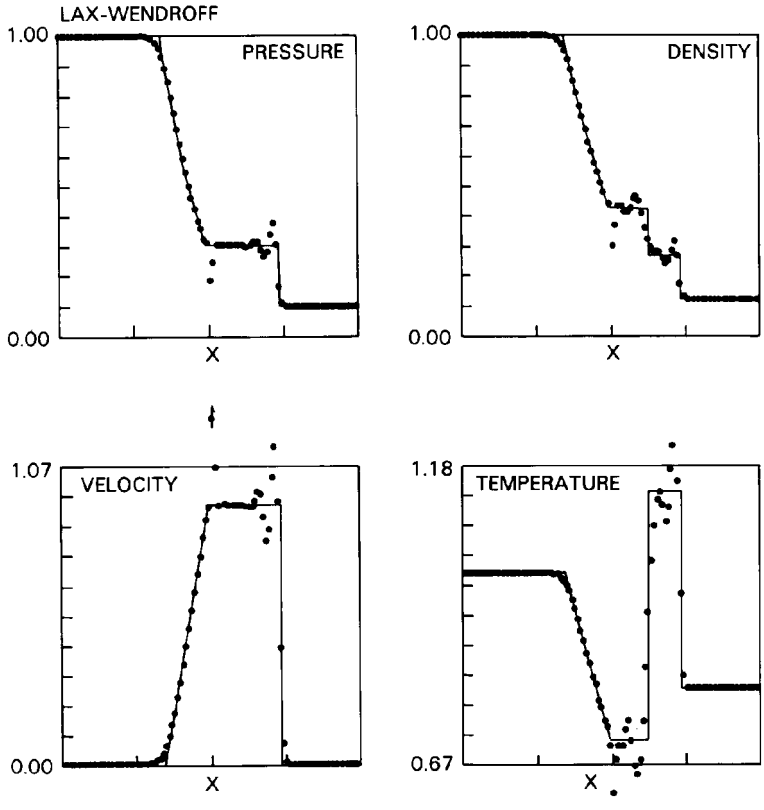

Figure 9.3. The Lax-Wendroff (MacCormack) method applied to the Riemann test problem. (Courtesy of S. Zalesak.)

a very short time has elapsed since bursting the diaphragm, so that the shock and contact surface are only eight or nine cells apart. In longer tests, the two nonlinear monotone methods would appear even better, relative to the Lax-Wendroff and Godunov methods.

The Lax-Wendroff method, shown in Figure 9.3, was programmed without introducing artificial viscosity. The absence of first-order damping shows large unphysical fluctuations in the variables at almost all of the discontinuities. At the contact discontinuity, where the velocity and pressure should be constant and continuous, the Lax-Wendroff solution has unacceptably large oscillations. Because the method is second order, however, the locations of the discontinuities are rather well represented and the solution in the rarefaction is quite close to the theoretical solution.

Methods that make explicit use of physical and analytical insights into the behavior of the coupled system, in general, do better than methods that do not. The fourth-order flux-corrected transport method used to calculate Figure 9.4 is basically a Lax-Wendroff method with monotonicity enforced. This extra important piece of physics, when built directly into the method, improves the solution appreciably. Overshoots and undershoots are virtually gone, relative to those seen in Figure 9.3. Behind the shock, however, the temperature peak shows an error that arises from two sources. A relatively small part of the error arises because of the problem of flux synchronization, discussed in Section 8–4.

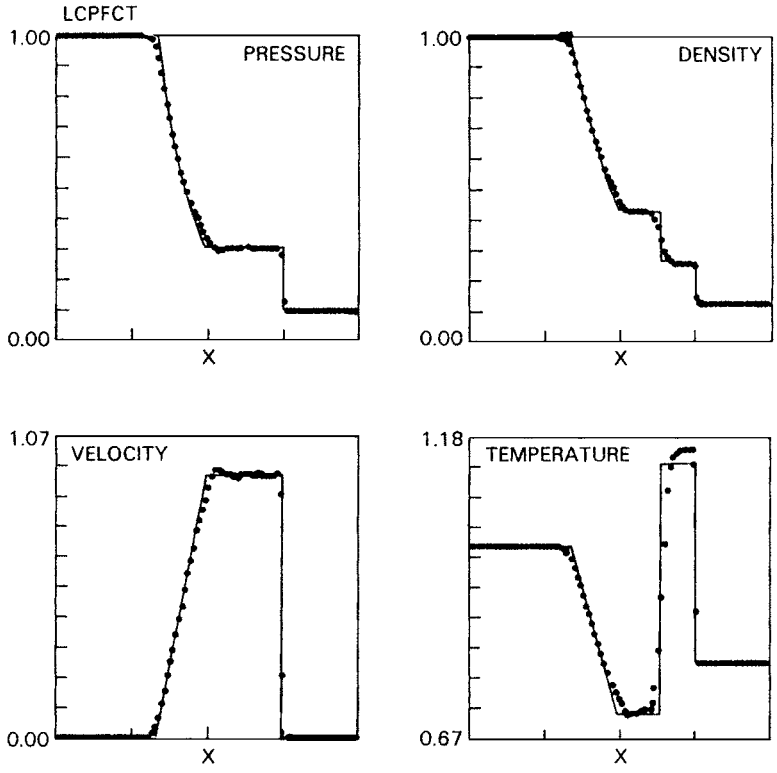

Figure 9.4. LCPFCT, a fourth-order version of flux-corrected transport, applied to the Riemann test problem. (Courtesy of S. Zalesak.)

The larger part of this error is residual and arises from initial transients that occurred in the first few timesteps when the diaphragm was broken. This error, which persists because the algorithm has very little numerical diffusion, dies out in long calculations.

Methods based on a Riemann solver use additional information about the actual solution relevant to this particular test problem, and such methods produce excellent results. The Godunov method does particularly well on the Riemann test problem considering it is only a first-order method. If used on the linear-convection problem shown in Figure 4.6, it performs as the donor-cell calculation, which is really not very good at all. There are no wild oscillations in the Godunov calculation, in contrast to the Lax-Wendroff calculation in Figure 9.3, because the Godunov method is monotone. The numerical diffusion is excessive in the region of the contact discontinuity and shock front. On a longer calculation of the same problem, the Godunov method shows appreciably more smearing at the contact discontinuity and rarefaction. Diffusion in the region of the shock front, however, is limited by the natural tendency of shocks to steepen and can be reduced somewhat when the calculation is performed at a higher Courant number. A Courant number of 0.9 gives a reasonably accurate representation of the shock front.

The last example, Figure 9.6, uses the SUPERBEE method (Roe 1985), which incorporates a solution of the Riemann problem, is a nonlinear monotone method with a flux-limiting procedure, and is second-order accurate. The quality of the solution is outstanding

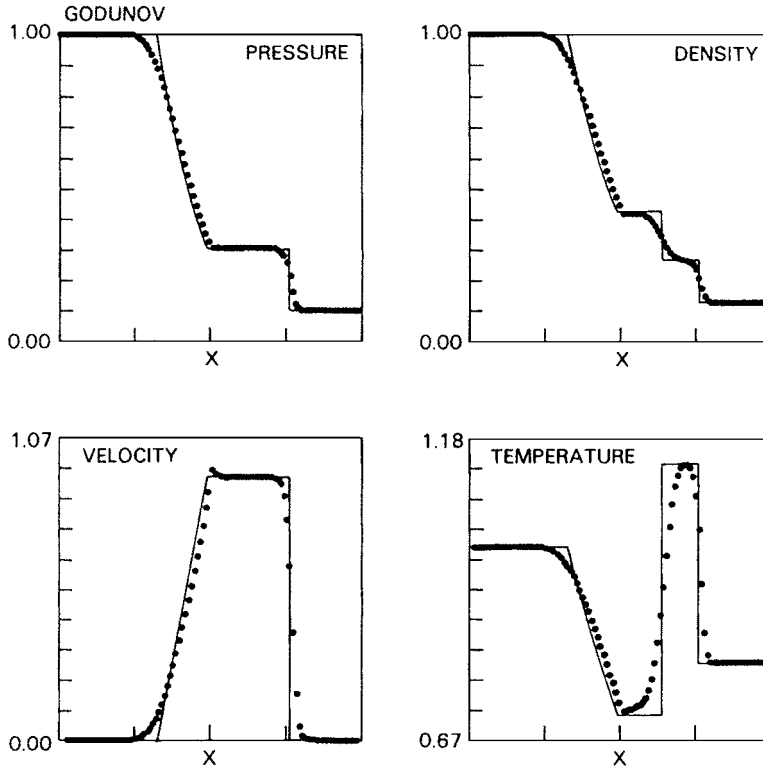

Figure 9.5. The Godunov method applied to the Riemann test problem. (Courtesy of S. Eidelman and S. Zalesak).

and the contact discontinuity is better resolved than it is in Figure 9.4. The method was designed to solve Riemann problems accurately, and it succeeds. For pure convection, however, it does not do as well as a fourth-order monotone method. In addition, it does not generalize to multidimensions or sliding grids, and it requires appreciable computation per timestep.

9–2.4. Shock Fitting and the Method of Characteristics

Flow characteristics were introduced earlier in this chapter. The *characteristic method* is the first of a class of closely related *shock-tracking* or *shock-fitting* methods for solving compressible flow problems. It is based on the idea of modeling an arbitrary flow as a series of isentropic regions separated by discontinuities. There are many good reviews of this method, for example, Liepmann and Roshko (1957) and the more recent discussion by Tannehill et al. (1997). This approach was extremely popular when computers were less powerful and even one-dimensional calculations were poorly resolved. These methods make maximum use of the limited data about the solution used in this representation, but have been generally replaced by more versatile, less expensive, and less complicated shock-capturing methods. The characteristic methods described briefly below are not applicable to incompressible flows where the sound speed is effectively infinite. For such cases,

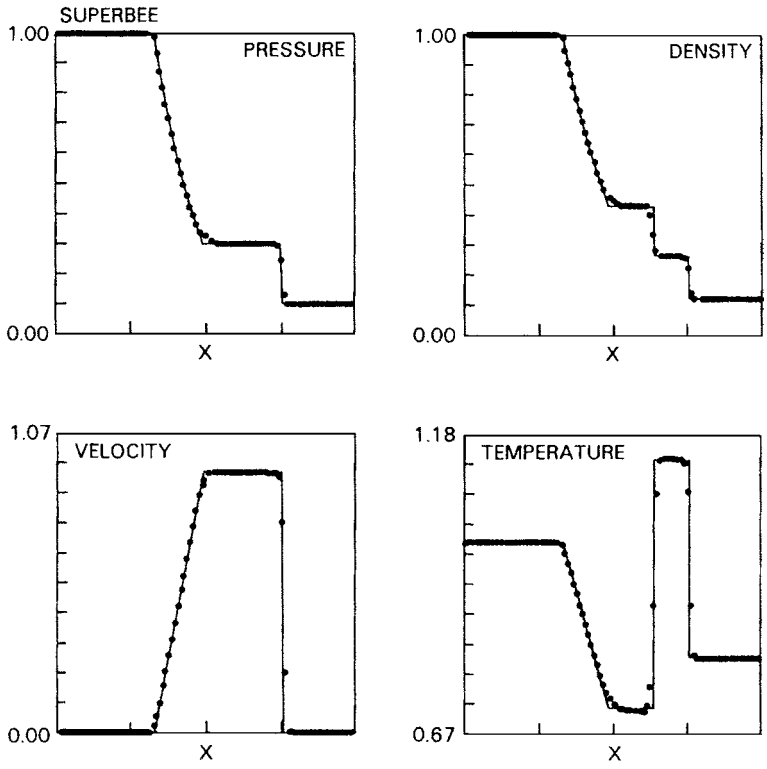

Figure 9.6. The SUPERBEE method applied to the Riemann test problem. (Courtesy of S. Zalesak.)

characteristic techniques reduce to algorithms that are very similar to the quasiparticle methods described later in this chapter.

In the characteristic method, it is necessary to distinguish six different kinds of points in the flow (see, for example, [Hoskin 1964]): points interior to isentropic regions, system boundaries, free surfaces, contact discontinuities, single shocks, and intersecting shocks. As discussed earlier in this chapter, a flow from a point has three characteristics along which information flows. The six types of point are labeled with coordinates x and t. Because time t, in general, varies from point to point, these methods do not need to work on a series of discrete time levels. To calculate the fluid properties at a new point on the intersection of characteristics running through two known points, only the physical quantities at the known points are needed. Thus it is necessary to store only an amount of information equivalent to describing one time level of a finite-difference scheme. In fact, some characteristic schemes use interpolation back onto a grid of points all at the same time level t. This permits the calculation to "march" from level to level.

Characteristic methods often can be formulated to take advantage of the physical content of a specific system of equations. For example, the equations of ideal magnetohydrodynamics (Kulikovsky and Lyubimov 1965) and ideal compressible fluid flow (Moretti and Abbett 1966) can be formulated and solved by the method of characteristics. Characteristic solutions to problems, where the method is applicable, are often excellent because the

discontinuities that arise naturally in these physical systems are followed individually and accurately.

On the other hand, characteristic methods are not generally applicable. The presence of even one diffusion term in the governing equations usually invalidates the use of characteristic methods because the characteristics of the composite model cease to exist! Diffusion also mixes regions of one entropy value with those of another, thereby requiring many more degrees of freedom to represent the smooth variation of the profile. A further drawback is that characteristic methods are quite complex and relatively inefficient for complicated and multidimensional flows. Special precautions are needed to describe a shock that forms from a steepening compression wave, a shock overtaking another shock, or a contact discontinuity forming when two shocks collide. All of these are situations in which discontinuities must be introduced or removed. In two dimensions, the characteristics are planar curves that intersect at moving points in x, y, t space. In three dimensions they are surfaces intersecting along moving curved lines. Thus in three dimensions, the method is even more complicated and the characteristics become progressively more and more difficult to follow as the flow proceeds in time. The logical complications of the characteristic method increase rapidly with the dimensionality and the number of discontinuities present, and the advantages become overwhelmed by the difficulties of accounting for all of the special cases that can arise (Hoskin 1964).

9–3. Methods for Incompressible Flows

No flow is really incompressible, certainly no chemically reacting exothermic gas or plasma flow. Incompressibility is a physical and mathematical limit that may be a good approximation when it is not necessary to resolve the sound waves in the Navier-Stokes equations. When the velocities are very low, fluid inertial effects and sound waves are usually unimportant. Nevertheless, the heat released by chemical reactions in the gas phase can cause significant expansion. Cooling due to radiation or thermal conduction, on the other hand, leads to local compression of the fluid. In general, $\nabla \cdot \mathbf{v} \neq 0$.

Ignoring sound waves is often justified when the flow changes so slowly that the finite sound speed seems infinite in comparison. The pressure then equilibrates very quickly throughout the entire system. There are even reacting flows that are "practically" incompressible. Water at high temperature and pressure still remains liquid, yet chemical reactions can occur in the solution. This is the basis for some waste-disposal systems. In the gas phase, trace reactants can undergo complex reactions and yet the fluid density and pressure remain constant. Nonetheless, even though incompressible flow algorithms are not generally applicable to reactive flows, they provide a useful basis for developing algorithms that can simulate slow, compressible flows, as described in Section 9–4.

The interaction between acoustics and pressure equilibration is subtle. Without the sound waves, the flow can display unphysical action at a distance. A force applied in one region, such as opening a door or valve, instantly induces motions elsewhere. In the real fluid, acoustic waves transmit these pressure pulses and the resulting gradients drive the motions. There are situations in which the action at a distance implied by infinite sound speed can lead to inconsistencies. Shear flow originating at a rigid splitter plate,

for example, might be treated as incompressible when the fluid velocity is low. Grinstein, Oran, and Boris (1990, 1991) showed that downstream vortex mergings generate acoustic pulses that move upstream to reinitiate the instability of the shear layer. If the numerical algorithms remove these pulses, or boundary conditions are selected that absorb or interfere with them, the simulated shear layer cannot sustain itself.

These inherent "action-at-a-distance" problems cause difficulties with numerical solutions of equations (9–1.1) through (9–1.3). The momentum and density continuity equations can be used to advance the solution, giving new fluid velocities. Incompressibility means, however, that the corresponding equation for energy (or pressure) is replaced by $\nabla \cdot \mathbf{v} = 0$ in concert with the continuity equation. The result is a Poisson equation, the solution of which can easily double the execution time, particularly for complex geometry. These concerns translate into design goals for algorithms to solve the incompressible-flow equations. The algorithms and grid representation should be simple for efficiency and ease of programming. Two- and three-dimensional implementations should be based on the same mathematical formulation. The same model should be able to solve unsteady as well as steady problems. The solution of Euler problems should be well approximated by taking the viscosity to zero. Whether or not all of the fluid-dynamic variables should be defined at the same positions is still a debated issue (see Löhner [1993] and Nicolaides [1993]).

In this section, we briefly describe three classes of approaches.

- *Pseudocompressibility methods*, sometimes called *artificial-compression methods*, artificially scale the pressure in the equations to reduce the disparity between the flow speed and the relatively high sound speed.
- *Divergence-projection methods*, or simply *projection methods*, correct the velocity field calculated by advancing the momentum equation. The correction is needed because an approximate pressure field is not usually fully consistent with $\nabla \cdot \mathbf{v} = 0$. To satisfy this constraint, an auxiliary elliptic equation is solved.
- *Implicit-pressure methods*, sometimes called *pressure-iteration methods* or *pressure-correction methods*, attempt to solve the exact problem accurately by solving a Poisson equation to determine the consistent pressure. Because direct methods for solving this equation are expensive and scale poorly as the number of grid points is increased, the resulting pressure equation is often solved iteratively.

In addition, Section 9–7 describes Lagrangian approaches called *vortex-dynamics methods* and *contour-dynamics methods*, where incompressibility is built in. In these approaches, the nonzero divergence is more difficult to deal with than incompressiblity, a situation which is quite the reverse of the grid-based solution methods for incompressible and slow flow problems.

Several numerical methods for solving incompressible flows are discussed in Roache (1982). The issues involved have led to a wide range of physical and mathematical approaches. The book edited by Gunzburger and Nicolaides (1993) contains a series of papers that provide in-depth discussion of the main computational issues and methods, including a range of gridding options. This is extended by Wilson, Demuren, and Carpenter (1998) and Pember et al. (n.d.).

9–3.1. Pseudocompressibility Methods

Chorin (1967) introduced *artificial-compressibility methods* as CFD algorithms in which the sound speed is artificially reduced to ease the timestep restriction in low-Mach-number flows. Buneman (1967) applied a similar idea to electrostatic plasma simulations (also see Birdsall and Fuss [1969] as reviewed by Birdsall and Langdon [1985]) by suggesting that the ratio of the ion to electron mass may be artificially reduced to lower the cost of computing electron-plasma oscillations. Buneman also suggested artificially reducing the speed of light in electromagnetic plasma simulations to reduce the disparity in timescales between electromagnetic-wave motion and particle motions. This basic idea has been extended to MHD for Alfvén waves by adding the electromagnetic-wave energy. With an artificially small value for the speed of light, the Alfvén speed does not become infinite as the gas density approaches zero (Boris 1970a,b).

Pseudocompressibility is really a modified representation of the incompressible fluid that retains a time-varying pressure term with a *pseudocompressibility factor* that produces low-speed acoustic waves. These pseudo-pressure fluctuations ensure that the flow remains nearly divergence free. The deviations from incompressibility represent an error that oscillates about zero and arises from the slightly incorrect equations being solved. A main advantage of pseudocompressibility is that solutions can be computed even with explicit integration algorithms of the type described earlier for fast flows. The time derivative of the pressure can be added to the continuity equation or by scaling the existing energy equation that would be used for a fully compressible model. For example, consider an equation of the form

$$\frac{1}{\beta}\frac{\partial P}{\partial t} = \frac{\partial \rho}{\partial t} = -\nabla \cdot \rho\, \mathbf{v}, \tag{9–3.1}$$

(see Bruel, Karmed, and Champion [1996]). Even though $\nabla \cdot \mathbf{v}$ should be zero, it is not strictly enforced in this equation. The method depends on the pseudodivergence to drive the velocity self-consistently and dynamically towards the divergence-free constraint.

In equation (9–3.1), β is a dimensional coefficient that scales the pseudopressure change. If the pseudosound speed c_{ps} is set equal to a multiple f of the maximum flow speed in the system $V_{\max}$, so that $c_{ps} = f V_{\max}$, the stagnation pressure is $P = f^2 \rho V_{\max}^2/2$. When $V_{\max} = 1$ m/s and $f = 10$, that is, when the pseudosound speed is $c_{ps} = 10$ m/s, the pseudopressure is a thousandth of atmospheric. The overall pressure is reduced to decrease the timestep, but the pressure differences required to drive fluid accelerations that maintain incompressibility must be unchanged. Thus the relative pressure and density fluctuations are magnified by the square of f.

Pseudocompressibility has been extended to slow chemically reactive flows. Bruel et al. (1996) recommend a pseudocompressibility factor of about eight based on the flow velocity in the burned products. Since the true sound speed is highest in the combustion products, the pseudo-Mach number $f\ M_a \approx 0.3$. This type of approach may be used when density variations are solely induced by the temperature change between two fixed states. It cannot be used even at constant pressure when density variations are large. In the general case, the pseudopressure can be found from an abstracted version of the energy

equation

$$\frac{\partial P}{\partial t} = -\beta \nabla \cdot P\,\mathbf{v}, \tag{9–3.2}$$

where now β is a dimensionless number. The density is advanced by the full continuity equation to ensure conservation of mass, even though the divergence *should* be zero. When the energy or pressure equation from a fully compressible model is modified, the gas temperature is, in effect, being reduced. If the correct temperature is then needed to compute the effects of some other process, such as chemical reactions, the reduced temperature can be multiplied by the scaling factor.

Pseudocompressibility algorithms are attractive approaches because they can be added to existing compressible fluid dynamics models with very little change. The problem is that they lead to a number of physical and numerical errors that must be understood to be evaluated. The nature of the approximation means no simple condition, such as a convergence criterion, can be used to examine the errors. Since the pseudosound waves travel slower than true sound waves, timing for acoustic-coupled effects occurs more slowly. Other consequences are potentially more serious. The resulting flow has a pseudo-Mach number, based on the actual flow speed and the reduced speed of the sound waves, that may be orders of magnitude larger than the real Mach number. If the velocity anywhere approaches or exceeds c_{ps}, completely unphysical shocks form, changing small quantitative errors into large qualitative ones.

Density and pressure fluctuations are also increased unphysically. In the pseudocompressible approximation with a scaled Mach number of 0.3, density fluctuations of a few percent occur with the pressure fluctuations. These would be only one part in 10^5 or 10^6 in the actual fluid. If the fluid is reacting above 1,000 K, small temperature variations associated with real sound waves are of little consequence, but the 20–30 K variations associated with pseudosound waves could be significant. In pseudocompressible flows, the small pressure differences that drive small-opening leakage, turbulent pipe flows, and porous-medium flows can become overshadowed by the pseudocompressibility effects. The enhanced density fluctuations can also cause unwanted artificial buoyancy effects. The effective atmospheric scale height, for example, is reduced by orders of magnitude. The bottom line is that pseudocompressibility algorithms can be extremely useful, but care must be taken to avoid unphysical consequences in complex problems.

9–3.2. Projection Methods

Suppose that the momentum equation (9–1.3) is advanced from the old time $(\rho\mathbf{v})^o$ to the new time $(\rho\mathbf{v})^n$ by a standard algorithm that uses the best available estimate of the pressure, or perhaps no pressure term at all. The problem is that the provisional velocity, $\tilde{\mathbf{v}}^n$, computed from $(\rho\mathbf{v})^n$, may not exactly satisfy $\nabla \cdot \tilde{\mathbf{v}}^n = 0$. Some sort of a correction $\delta\mathbf{v}^n$ to $\tilde{\mathbf{v}}$ is required, so that $\mathbf{v}^n = \tilde{\mathbf{v}}^n + \delta\mathbf{v}^n$ satisfies $\nabla \cdot \mathbf{v}^n = 0$. There are a number of ways to effect the velocity correction. One is to iterate on the pressure and velocity in an effort to find a small correction to the velocity consistent with the density equation and $\nabla \cdot \mathbf{v} = 0$. This idea was first used in fluid dynamics in the *marker and cell methods* (Harlow and Welch 1965; Welch et al. 1966).

As discussed in Chapter 2, the velocity field $\mathbf{v}(x, y, z) = \mathbf{v}(\mathbf{r})$ at any instant can always be written as the sum of three terms,

$$\mathbf{v}(\mathbf{r}) = \nabla \times \boldsymbol{\psi}(\mathbf{r}) + \nabla\phi(\mathbf{r}) + \nabla\phi_p, \tag{9–3.3}$$

where the vector $\boldsymbol{\psi}$ is the stream function, ϕ the velocity potential, and ϕ_p is a particular velocity potential that may be added to the equation to ensure consistency with complex geometry boundary conditions or far field flow conditions. By ignoring the particular solution and taking the divergence and curl of equation (9–3.3), we can write

$$\nabla \cdot \mathbf{v} = \nabla \cdot (\nabla \times \boldsymbol{\psi}(\mathbf{r})) + \nabla \cdot \nabla\phi(\mathbf{r}) = \nabla^2\phi(\mathbf{r}) \tag{9–3.4}$$

and

$$\nabla \times \mathbf{v} \equiv \boldsymbol{\omega}(\mathbf{r}) = \nabla \times (\nabla \times \boldsymbol{\psi}(\mathbf{r})) + \nabla \times \nabla\phi(\mathbf{r}) = \nabla \times \nabla \times \boldsymbol{\psi}(\mathbf{r})). \tag{9–3.5}$$

These two terms are the inhomogeneous components of the *div-curl* problem. In general the div-curl problem (e.g., Bertagnolio and Daube 1997) refers to the reconstruction of a unique velocity field $\mathbf{v}$ from a given scalar divergence field $\nabla \cdot \mathbf{v}$ and a divergence-free, vector vorticity field $\boldsymbol{\omega}(\mathbf{r}) \equiv \nabla \times \mathbf{v}$.

For incompressible flow, $\nabla \cdot \mathbf{v}$ gives a Poisson equation for the velocity potential corresponding to the unwanted compression,

$$\nabla \cdot \mathbf{v} = \nabla^2\phi \approx 0. \tag{9–3.6}$$

This should be, but usually is not, identically zero. Thus projection methods are a special case of the general div-curl problem. Calculating ϕ to correct $\tilde{\mathbf{v}}$ forms the basis of the class of methods called *divergence-projection methods*, projection methods, or *fractional-step methods*. These methods can bypass the use of the pressure as an intermediate variable to find ϕ needed to satisfy $\nabla \cdot \mathbf{v} = 0$. This idea is expressed mathematically by saying that the correct velocity can be found by projecting the computed velocity onto a subspace in which $\nabla \cdot \mathbf{v} \equiv 0$. These methods have been developed by Chorin (1968) and Temam (1969), with more recent work by Donea et al. (1982), Patera (1984), and Kim and Moin (1985), among many others. Typically these methods solve evolution equations for the density, velocity, enthalpy, and species concentrations without requiring a specific solution for the pressure (Bell, Colella, and Glaz 1989; Bell and Marcus 1992; Pember et al. n.d.).

Projection methods are often used on staggered grids, as discussed in Section 6–2, so that the velocity components are defined at locations that naturally lead to simple definitions of the divergence, curl, and gradient difference operators. Such grids are a useful way to avoid certain numerical problems that appear as spurious modes in the solution. Another solution for this problem, when a regular, unstaggered grid is used, is to add moderate dissipation to damp the spurious modes (Löhner 1993).

The existence of a body of theory for projection methods, with its close attention to errors and convergence, has made these methods attractive for incompressible fluid-dynamics problems. Nonetheless, difficulties and errors do arise. Even though the numerically generated errors in the divergence are corrected after they have been created, there is still the possibility of growing secular errors. Even though $\nabla \cdot \mathbf{v} = 0$, nothing has been done to

correct the errors in $\mathbf{v}$ arising from the curl of the stream function $\nabla \times \boldsymbol{\psi}$. The results are possible growing errors in the vorticity ω. The specter of action-at-a-distance problems associated with infinite-speed sound waves is also inherent in the method. Consequently, the solutions are sensitive to the boundary conditions, even those at infinity. The results are also sensitive to errors that might arise if the solution of the elliptic equations is not accurate enough. In all cases, obtaining high-accuracy solutions of the elliptic equations can be time consuming and complicated to program for complex geometries.

Many applications require knowing the pressure. When a projection method is extended to find a consistent, improved approximation to the actual pressure, it can become indistinguishable from the implicit pressure formulations described next. Other applications have a nonzero divergence in some portions of the domain. This happens, for example, in the reaction zones of flames where heat release causes considerable expansion. In this case, the projection method is further complicated because the projection is no longer on a velocity subspace of zero divergence. Thus there is further potential for inconsistency and secular errors. The slow-flow algorithms discussed in Section 9–4 are methods for allowing nonzero divergence. In Section 9–3.4, we describe a simple implicit algorithm which may be classified either as a projection method or as a pressure-iteration method, depending on how it is implemented.

9–3.3. Implicit Pressure Formulations

Solving the complete flow problem requires finding the pressure variations in space and time that maintain $\nabla \cdot \mathbf{v} = 0$. Pseudocompressibility and projection methods enforce incompressibility directly and allow a flow computation to proceed without necessary reference to the physically correct pressure. There are, however, many types of problems in which accurate prediction of the pressure itself is required. In the high-pressure water reactor, for example, the pressure directly controls bubble formation and cavitation. In gravitationally stratified fluids, the pressure gradients balance gravity. In buildings and porous media, small pressure differences control leakage from one region to another. Therefore, solving the elliptic equation for the pressure is often necessary.

The need for knowing an accurate pressure in some types of incompressible problems has led to a class of methods called *pressure-iteration methods*. The name refers to the fact that the implicit pressure equation, particularly with a complex flow geometry, is difficult to solve directly, and thus iteration of some kind is needed. The general formulations are based on the usual procedure of substituting the density equation into the momentum equation, and somehow imposing the zero-divergence velocity criterion on the problem. Often a fractional timestep is used in the process.

An advantage of the implicit pressure formulation is that the actual, local equation of state is included in the solution procedure. This helps to keep the evolving density and pressure fully coupled and controls some forms of error. The result is that the accuracy of the solution of the associated elliptic equation can often be very greatly reduced. A few iterations at each timestep, using a pressure extrapolated from the previous timestep, is often adequate. The representation and data structures are still important, because implicit pressure formulations also are subject to spurious modes. Thus the concepts of staggered or covolume grids are still needed. An early application of these methods to unstructured

Lagrangian grids of reconnecting triangles and tetrahedra was developed by Fritts and Boris (1979), who were able to avoid the spurious modes by appropriately determining a way in which to define the variables as either vertex, or cell-center, or cell-interface quantities.

A number of algorithms have been developed around implicit pressure formulations. Many of these deal with compressible flows, so that $\nabla \cdot \mathbf{v}$ need not be zero. This is the case for the slow flows discussed in Section 9–4. For these flows, the divergence is not zero, but the sound waves still must somehow be filtered out of the system or otherwise be stabilized.

9–3.4. A Method for Incompressible Flows

There is a rather straightforward, three-step method (Chorin 1968) that provides a good introduction to programming algorithms for incompressible flows. This method may be variously called a projection method or a pressure-correction method. The starting points are equations (9–1.2) and (9–1.3a). For simplicity, we assume $\mathbf{f}_e = 0$. Then we define an intermediate velocity, denoted by $\mathbf{v}^*$, so that

$$\frac{\mathbf{v}^* - \mathbf{v}^n}{\Delta t} = -\mathbf{v}^n \cdot \nabla \mathbf{v}^n + \frac{\mu}{\rho} \nabla^2 \mathbf{v}^n \tag{9–3.7}$$

and the velocity at the new time $n+1$ is

$$\frac{\mathbf{v}^{n+1} - \mathbf{v}^*}{\Delta t} = -\frac{1}{\rho} \nabla P^{n+1}. \tag{9–3.8}$$

Imposing $\nabla \cdot \mathbf{v}^{n+1} = 0$, we find

$$\nabla^2 P^{n+1} = \frac{1}{\rho\, \Delta t} \nabla \cdot \mathbf{v}^*. \tag{9–3.9}$$

The three-step process is to (1) compute $\mathbf{v}^*$ from equation (9–3.7), (2) compute P^{n+1} from equation (9–3.9), and finally (3) compute $\mathbf{v}^{n+1}$ from equation (9–3.8). Many extensions of this are possible, including using the method on various types of staggered grids (so that the method becomes very much like the marker-and-cell (MAC) method) and unstructured grids (see, for example, Morgan and Peraire [1998]), and using a provisional pressure in equation (9–3.7) (see Tannehill et al. [1997]).

9–4. Methods for Slow Flows

In the slow-flow regime, the fluid and reaction fronts are moving much slower than the speed of sound, but compressibility can still be extremely important. A laminar flame is an example of such a slow flow. The flame consumes combustible gases slowly enough for the pressure to stay essentially constant. Across the flame, however, the density can change by factors of four to ten because of heat release and the accompanying expansion. Neither the density change nor the fluid displacement associated with the expansion can

be ignored, even though the pressure fluctuations that drive the local expansion are very small. Furthermore, transient expansions generate vorticity baroclinically, through the term $\nabla P \times \nabla \rho$, even though the pressure is essentially constant.

The numerical issue for slow flows is whether the time integration algorithm should treat sound waves explicitly, implicitly, or even asymptotically. Explicit algorithms, no matter how efficient computationally, become expensive when the maximum flow speed in the computational domain drops below 10 percent of the sound speed. Implicit methods usually involve expensive, iterated solutions of an elliptic equation. These additional computations can take as long as five or ten timesteps of an explicit algorithm, and they can become prohibitive in complex geometry. Alternate approaches include asymptotic slow-flow and pseudocompressibility methods that analytically rescale the properties of the fluid equations. In the material presented in Section 9–4.1, we describe several implicit, Eulerian approaches that we have found useful.

9–4.1. The Slow-Flow Methods

Slow-flow methods are hybrids of asymptotic and implicit approaches for solving the coupled continuity equations for low-speed flows. These methods include the algorithms by Ramshaw and Trapp (1976), Jones and Boris (1977, 1979), Rehm and Baum (1978), Rehm, Baum, and Darcy (1982), and Paolucci (1982). The Jones-Boris algorithm was originally designed for multidimensional flame simulations. The equations solved are the density continuity equation, the various species continuity equations with reaction kinetics source terms, and the convective vorticity equation, equation (9–1.7). The elliptic relations for the velocity potential, equation (9–3.4), and the stream function, equation (9–3.5), are solved to construct the instantaneous velocity.

The final equation used is the isentropic pressure equation with source terms,

$$\frac{\partial P}{\partial t} + \mathbf{v} \cdot \nabla P = -\gamma P \, \nabla \cdot \mathbf{v} + \left. \frac{\partial P}{\partial t} \right|_{\text{chem}} - (\gamma - 1)[\nabla \cdot \mathbf{q} - \mathbf{v} \cdot (\nabla \cdot \mu \nabla \mathbf{v})]. \tag{9–4.1}$$

The heat flux q, defined in equation (2–1.6) in chapter 2, includes thermal conduction and molecular diffusion. The effects of chemical reactions are included through the species production terms $\{P_i\}$ and loss terms $\{L_i\}$ in the species density equations. The term $\partial P / \partial t|_{\text{chem}}$ arises from the change in enthalpies due to the chemical reactions and appears when we convert the energy equation to a pressure equation. The physical velocity divergence to lowest order in fluctuations of the pressure is defined from equation (9–4.1),

$$\gamma P \, \nabla \cdot \mathbf{v} \approx \left\langle \left. \frac{\partial P}{\partial t} \right|_{\text{chem}} - \frac{\partial P}{\partial t} \right\rangle + (\gamma - 1) \left[\nabla \cdot \mathbf{q} - \mathbf{v} \cdot (\nabla \cdot \nu \nabla \mathbf{v}) + \sum_i h_i \rho_i \mathbf{v}_{di} \right]. \tag{9–4.2}$$

This is a zeroth-order expression for $(\nabla \cdot \mathbf{v})$ that neglects the $\mathbf{v} \cdot (\nabla P)$ term and averages over the time derivative. In certain fluid dynamic problems, it is sometimes necessary

to solve equation (9–4.2) to first order in pressure fluctuations to compute the vorticity generation self-consistently.

One interesting feature of equation (9–4.2) is the first term on the right side, the average pressure change in the system from external energy addition or compression. This is written as the average of the $\partial P/\partial t$ terms in equation (9–4.1). A net addition of heat to the system changes the average pressure with time. In a closed container, however, this average heating can lead to no net divergence and therefore is subtracted from equation (9–4.2) to yield the correct *local* value of $\nabla \cdot \mathbf{v}$.

The solution to these slow-flow equations no longer includes sound waves and thus the formulation cannot treat shocks. The method still incorporates conduction, compression, and rarefaction as long as they evolve slowly compared to the propagation time of sound waves across the system. Two elliptic equations, for ϕ and another for $\boldsymbol{\psi}$, are solved as indicated in equations (9–3.4) and (9–3.5). To lowest order in the ratio of time scales, the pressure cannot vary in space. Pressure can, however, vary globally on the time scale of the energy release.

There is an advantage to this type of method despite the need to solve two elliptic equations: the way in which the nonlinear continuity equations are advanced is left unspecified, so that optimal convection algorithms can be chosen to solve them. For example, either Eulerian or Lagrangian finite-difference methods can be used.

Jones and Boris used an Eulerian FCT algorithm similar to that described in Section 8–3. Any good, monotone algorithm would be appropriate. They implemented the model on a staggered grid similar to that shown in Figure 6.4. The density equation has ρ defined on the grid and $\mathbf{v}$ defined at intermediate locations. Thus $(\nabla \cdot \mathbf{v})$ can be evaluated directly on the same grid as ρ, and $(\nabla \times \mathbf{v})$ is defined at the corners of the cells where $\boldsymbol{\psi}$ is defined. When various fluxes are required, such as for calculating diffusion velocities, some of the quantities need to be averaged in space. This staggered-grid representation maintains second-order accuracy for uniform or slowly varying cell sizes while defining all the vector identities consistently and ensuring there are no spurious modes. Nevertheless, complex geometry is difficult to represent, and the two solutions of the elliptic equations that are required are a serious computational drawback.

Ramshaw and Trapp (1976) were interested in a multiphase-flow problem in which they needed to represent the surface between a liquid and a gas. Their algorithm is similar to the slow-flow algorithm presented above with an important difference. They derive an expression for the velocity divergence, but do not decouple it from the sound waves in the system. Because of this coupling, they use an implicit formulation of the momentum and density equations that precludes using a positivity-preserving convection algorithm.

Rehm and Baum (1978) and Rehm et al. (1982) derived the hierarchy of slow-flow equations from a general perspective. They present a systematic expansion procedure in which the only effects of compression allowed are the asymptotic long-time changes in density that result from local heating and cooling. The first-order term in their expansion of $(\nabla \cdot \mathbf{v})$ is the expression given in equation (9–4.2). They originally presented their derivation in the context of primitive variables, and later in the vorticity formulation. Paolucci (1982) discussed the physical assumptions inherent in slow-flow methods, the scaling of the physical terms in the fluid equations, and the relation of the slow-flow equations to other approximations.

More recently, it has been recognized that primitive variable formulations can properly advance the vorticity without solving the Poisson equation for the stream function, provided the difference operations properly mirror the vector identities $\nabla \cdot \nabla \times \mathbf{A} = 0$ and $\nabla \times \nabla \phi = 0$. Then the momentum equation properly conserves the circulation for a wide range of difference approximations. Thus it is attractive to generalize the projection methods of Section 9–3.2 to the case where the divergence is not zero but given by equation (9–4.2). The need for high accuracy in the elliptic-equation solution for incompressible projection methods is not relaxed, nor is the need for absolute consistency of the boundary conditions and the elliptic solution. Using a projection method, however, only one elliptic equation must be solved.

Slow-flow methods work well when acoustic fluctuations are not important to the fluid dynamics, that is, for constant-pressure, subsonic flows. In fact, sound waves are simply not included in these methods. Slow-flow methods and the closely related projection methods break down when the kinetic energy of the flow becomes comparable to the internal energy.

9–4.2. An Implicit Flux-Corrected Transport Algorithm

In many reactive-flow problems, such as those involving material ablation or flames, pressure variations in the form of sound waves are important even though the fluid profiles themselves vary slowly. In these problems, an explicit integration algorithm is too expensive, but a slow-flow expansion based on constant pressure is not valid. Also, in atmospheric flows, in turbulence modeling for chemically reactive systems, and for problems involving Rayleigh-Taylor instabilities, the pressure fluctuations interact with the local density gradients to generate vorticity. In these cases, the acoustic waves may be treated implicitly as long as shocks are not present. For this reason, and because it is expensive to solve two elliptic equations very accurately, implicit and semi-implicit methods often supplant asymptotic slow-flow methods. This is especially the case for reactive flows.

The fundamental idea of *BIC* (Patnaik et al. 1987) is to find an implicit pressure-correction algorithm that could be embedded in a high-order explicit method, thus circumventing the CFL condition on the sound speed by adding one implicit elliptic equation. BIC is based on the analysis of Casulli and Greenspan (1984), who showed that it is not necessary to treat all terms in the gas dynamic equations implicitly to use longer timesteps than allowed by explicit stability limits. Only the two specific terms that force the numerical stability limit need to be solved implicitly. The Patnaik versions of BIC use FCT to solve continuity equations. Nonetheless, any of the good, nonlinear monotone convection algorithms described in Chapter 8 should maintain the steep gradients accurately enough.

There are two versions of BIC now in use, the original approach (Patnaik et al. 1987) and the new approach (Patnaik, n.d.). The basic BIC is a sequence of five steps.

1. Perform an explicit predictor step that solves coupled continuity equations to determine new values of density ρ^n, species densities $\{n_i\}$, predicted momentum $\rho^n \widetilde{\mathbf{v}}$, and energy $\widetilde{E}$ densities based on nonlinear convection using FCT.
2. For an imperfect gas, use the thermal equation of state to compute a provisional internal energy, $\epsilon^* = H(P^o, \{n_i^n\}) - P^o$, using the old pressure and new densities. For a perfect gas, use $\epsilon^* = P^*/(\gamma - 1)$. The provisional total energy density is

$E^* = \epsilon^* + \frac{1}{2}\rho^n \widetilde{\mathbf{v}}^2$. This relationship between energy and pressure is used to derive the elliptic pressure equation (given below).

3. Solve this elliptic equation for the new pressure P_g^n. The new BIC uses a *gauge pressure*, $P_g = P - P_\infty$, to avoid round-off and truncation errors. (This is *not* a pressure increment from the previous timestep, as it was in the original version of BIC.)
4. Update the predicted momentum using the new implicit pressure to give $\rho^n \mathbf{v}^n$. This also makes the new kinetic energy available.
5. For an imperfect gas, use the thermal equation of state to compute the new energy density $E^n = H(P^n, \{n_i^n\}) - P^n + \frac{1}{2}\rho^n \mathbf{v}^{n2}$, using the new pressure and chemical species. When a perfect gas equation of state is used, $E^n = P^n/(\gamma - 1) + \frac{1}{2}\rho^n \mathbf{v}^{n2}$.

In the original implementation, E was an independent variable, as was $\{n_i\}$, ρ, and $\rho\mathbf{v}$, and pressure and temperature were derived. In the new version, E (and ϵ) are derived quantities. The new P_g is computed directly, not as a correction to the old pressure. This makes the new version less sensitive to small errors and fluctuations, and so it is less difficult to solve the elliptic equation. In the reactive-flow equations, the change in internal energy is mainly due to the change in all of the number densities, $\{\Delta n_i\}$, and not so much due to changes in pressure, which is virtually constant. Now $\{\Delta n_i\}$ and P_g are thought of as the independent variables, and these are used to find T, ϵ, and E. Thus the new BIC algorithm is more robust and less sensitive. Because of the advantages of the new formulation, we describe that in more detail now.

The basic equations for the new BIC are for density, momentum, energy density, and species densities,

$$\frac{\partial \rho}{\partial t} = -\nabla \cdot \rho \mathbf{v} \tag{9–4.3}$$

$$\frac{\partial \rho \mathbf{v}}{\partial t} = -\nabla \cdot \rho \mathbf{v}\mathbf{v} - \nabla P_g \tag{9–4.4}$$

$$\frac{\partial E}{\partial t} = -\nabla (E + P) \cdot \mathbf{v} \tag{9–4.5}$$

$$\frac{\partial n_i}{\partial t} = -\nabla \cdot n_i \mathbf{v}. \tag{9–4.6}$$

Since the gradient of the reference pressure P_∞ is zero, the momentum equation is unchanged with P_g, but P must appear in the equation (9–4.5) for the energy. This modification reduces difference-of-large-number errors in slow flows where the pressure fluctuations are orders of magnitude smaller than the overall pressure.

These equations are discretized in time to give

$$\frac{\rho^n - \rho^o}{\Delta t} = -\nabla \cdot \rho^o \mathbf{v}^o, \tag{9–4.7}$$

$$\frac{\rho^n \mathbf{v}^n - \rho^o \mathbf{v}^o}{\Delta t} = -\nabla \cdot \rho^o \mathbf{v}^o \mathbf{v}^o - \nabla P_g^n, \tag{9–4.8}$$

$$\frac{E^n - E^o}{\Delta t} = -\nabla \cdot (E^o + P^o)\mathbf{v}^n. \tag{9–4.9}$$

The corresponding continuity equation for the chemical species i is

$$\frac{n_i^n - n_i^o}{\Delta t} = -\nabla \cdot n_i^o \mathbf{v}^o. \tag{9–4.10}$$

The energy and chemical-species number densities in equations (9–4.9) and (9–4.10) will have already been updated using an operator-split reaction-rate integration.

Now define an intermediate momentum $\rho^n \widetilde{\mathbf{v}}$ that neglects the pressure gradient term in equation (9–4.8),

$$\frac{\rho^n \widetilde{\mathbf{v}} - \rho^o \mathbf{v}^o}{\Delta t} = -\nabla \cdot \rho^o \mathbf{v}^o \mathbf{v}^o. \tag{9–4.11}$$

Subtracting equation (9–4.11) from equation (9–4.8) gives

$$\mathbf{v}^n = \widetilde{\mathbf{v}} - \frac{\Delta t}{\rho^n} \nabla P_g^n. \tag{9–4.12}$$

Substituting equation (9–4.12) into equation (9–4.9), gives an equation for the energy in terms of the undetermined new pressure,

$$\frac{E^n - E^o}{\Delta t} = -\nabla \cdot (E^o + P^o)\widetilde{\mathbf{v}} + \Delta t \, \nabla \left(\frac{E^o + P^o}{\rho^n} \right) \cdot \nabla P_g^n. \tag{9–4.13}$$

As with the momentum in equation (9–4.12), define an intermediate energy $\widetilde{E}$ from equation (9–4.13) (again neglecting the pressure term)

$$\frac{\widetilde{E} - E^o}{\Delta t} = -\nabla \cdot (E^o + P^o)\widetilde{\mathbf{v}}. \tag{9–4.14}$$

The FCT algorithm and integration package, described in Sections 8–4.2 and 8–4.5, is used to integrate equations (9–4.7), (9–4.10), (9–4.11), and (9–4.14) one timestep to provide the updated variables on the left-hand side. Now substitute equation (9–4.14) into equation (9–4.13), giving

$$E^n = \widetilde{E} + (\Delta t)^2 \, \nabla \left(\frac{E^o + P^o}{\rho^n} \right) \cdot \nabla P_g^n. \tag{9–4.15}$$

Equation (9–4.15) is the desired implicit equation for the pressure, once the relation between the energy density and the pressure is incorporated.

Now consider some state, designated by E^* and P_g^*, close to the desired new timestep values with the same total number density N^n. Then if, for simplicity, we assume an ideal gas law and equation of state, we can write

$$E^n = E^* + c_v \, \Delta T \tag{9–4.16}$$

$$P_g^n = P_g^* + N^n \, k \, \Delta T. \tag{9–4.17}$$

When the equation of state is very nonlinear, a linear expansion can be made around the new state, or an iteration procedure used to find the new state. The quantities E^* and E^n both include the kinetic energy, which is not yet known because $\mathbf{v}^n$ is as yet undetermined. Equation (9–4.16) has the kinetic energy implicitly on both sides of the equation, so it cancels. Then eliminating ΔT,

$$E^n = E^* + \frac{c_v}{N^n k}\left(P_g^n - P_g^*\right) \tag{9–4.18}$$

and using the relations $c_p - c_v = Nk$ and $\gamma = c_p/c_v$, we find

$$E^n = E^* + \frac{1}{\gamma - 1}\left(P_g^n - P_g^*\right). \tag{9–4.19}$$

Substituting equation (9–4.19) into equation (9-4.15) gives

$$E^* + \frac{1}{\gamma - 1}\left(P_g^n - P_g^*\right) = \widetilde{E} + (\Delta t)^2 \nabla\left(\frac{E^o + P^o}{\rho^n}\right) \cdot \nabla P_g^n. \tag{9–4.20}$$

Gathering terms with P^n to the left, we find the new BIC implicit equation for the gauge pressure,

$$\frac{1}{\gamma - 1} P_g^n - (\Delta t)^2 \nabla\left(\frac{E^o + P^o}{\rho^n}\right) \cdot \nabla P_g^n = \widetilde{E} - E^* + \frac{P_g^*}{\gamma - 1}. \tag{9–4.21}$$

Now set $P_g^* = P_g^o$ and find E^* from the thermal equation of state,

$$E^* = h\left(P^o, \left\{n_i^n\right\}\right) - P^o + \frac{1}{2}\rho^n \widetilde{\mathbf{v}}^2. \tag{9–4.22}$$

In this formulation, as opposed to the original formulation, we have ignored the implicitness factor for simplicity, setting $\theta = 1$.

When BIC-FCT is used in multidimensional problems, the five-step procedure given above is used (Patnaik n.d.). Advancing the timestep in the explicit part of the procedure, equations (9–4.7), (9–4.10), (9–4.11), and (9–4.14), requires only a single step instead of the half-step, whole-step procedure described in Sections 8–3 and 8–4. The explicit convection can be one-dimensional, direction-split, or fully multidimensional. For the method to work, however, the elliptic pressure equation must be solved in multidimensional form, and this is a substantial part of the computational effort at each timestep. In one dimension, the finite-difference form of the pressure-difference equation can be solved efficiently using standard tridiagonal methods in $\mathcal{O}(N)$ operations, where N is the number of grid points. In two or three dimensions, it is important to use an efficient elliptic solver, preferably one that is not limited to specific types of problems with specific boundary conditions. Methods for solving tridiagonal matrices and elliptic equations are described briefly in Chapter 10.

Some General Comments

The BIC-FCT approach is more general than the slow-flow method, and therefore it is useful as a single accurate method in the range from nearly incompressible to subsonic

flows. It also seems easier and better to use for faster flows because it allows acoustic modes. Patnaik has shown that when the timestep is reduced to the explicit limit determined by the sound speed, this algorithm does a reasonable calculation of the Riemann shock problem.

An important point to note is that the cost of an implicit timestep is about the same as for an explicit timestep. As described in Chapter 8, the explicit calculations require both a half-step prediction and whole-step correction procedure each timestep. BIC-FCT does not split the timestep, but requires instead one elliptic solution. If that solution is efficient enough, performing it and the auxiliary calculations take roughly the same computing time as the half-timestep calculations in the explicit FCT approach. The implicitness parameter, θ, plays an important role whenever sound waves and pressure oscillations are important. Patnaik showed that damping is negligible for long wavelengths and timesteps when $\theta = 0.5$ and the Courant number for the sound speed is less than two. Dispersion and damping become important for Courant numbers greater than five, irrespective of the value of θ. When sound waves are not important, θ can be set to unity as in the exposition above.

9–4.3. Several Implicit Algorithms

The slow-flow method, and the implicit adaptation of FCT discussed above, are only two of many approaches to solving fluid equations in the slow-flow regime. Most of the other methods in use now for this regime are either projection methods or are fully implicit. For these we recommend standard texts, such as Tannehill et al. (1997) and Hirsch (1990) for more detailed descriptions. Here a brief description of several of these methods gives a flavor of the possibilities.

Block-Implicit Methods

Block-implicit methods were developed by Lindemuth and Killeen (1973), Briley and McDonald (1975, 1977, 1980), Beam and Warming (1976, 1978), and Warming and Beam (1977) for solving compressible, time-dependent Euler and Navier-Stokes equations. These methods are generally first or second order in time and very stable. Here we briefly describe the classic Beam-Warming method.

The starting point for the Beam-Warming method is the conservation equations for density, momentum, and energy written in matrix form in terms of a composite vector variable $\mathbf{u}$, as we did for the Lax-Wendroff algorithm in Section 8–4.1. Here the viscous terms are kept in the full Navier-Stokes equations. The solution vector $\mathbf{u}$ is advanced in time by the formula

$$\Delta^n \mathbf{u} \equiv \mathbf{u}^{n+1} - \mathbf{u}^n = \left(\frac{\theta_1 \Delta t}{1+\theta_2}\right)\frac{\partial}{\partial t}\Delta^n \mathbf{u} + \left(\frac{\Delta t}{1+\theta_2}\right)\frac{\partial}{\partial t}\mathbf{u}$$
$$+ \left(\frac{\theta_2}{1+\theta_2}\right)\frac{\partial}{\partial t}\Delta^{n-1}\mathbf{u} + \mathcal{O}\left[\left(\theta_1 - \frac{1}{2} - \theta_2\right)(\Delta t)^2 + (\Delta t)^3\right]. \qquad (9\text{–}4.23)$$

The proper choice of θ_1 and θ_2 produces standard difference methods. For example, $\theta_1 = \theta_2 = 0$ gives an explicit first-order Euler method. The values $\theta_1 = 1$ and $\theta_2 = 0$ give an implicit Euler method, also first order in time. The most common method is $\theta_1 = 1$, $\theta_2 = \frac{1}{2}$, the three-point backward implicit method that is second order in time.

The procedure is to substitute the equations for **u** into equation (9-4.23) and linearize the nonlinear terms. For example, we find terms of the form

$$\Delta^n \mathbf{f}_y = \mathbf{A}_x \Delta^n \mathbf{u} + \mathcal{O}\left[(\Delta t)^2\right], \tag{9–4.24}$$

where $\mathbf{f}_y$ is as defined in Section 8–4.1 and **A** is a Jacobian matrix. The resulting set of equations is a block-tridiagonal matrix system solved by ADI methods, discussed in Section 10–3.

Obayashi and Fujii (1985) developed a block-implicit method for the multidimensional Navier-Stokes equations using the flux-vector splitting method described previously in Section 9–2.2. They factored the fluxes into the right- and left-moving components, and treated each implicitly. Their equations resulted in bidiagonal instead of tridiagonal or pentadiagonal solutions in the appropriate directions. The different spatial directions are again treated by ADI methods. This method is first order and there is no monotonicity correction attempted. The general technique used here for simplifying the implicit calculations is useful.

Implicit Methods with Riemann Solvers

A number of the explicit methods that include Riemann solvers have been made implicit. These include the MUSCL method, extended by Mulder and Van Leer (1985), the PPM method, extended by Fryxell et al. (1986), and the implicit TVD method by Yee and Harten (1985) and Yee, Warming, and Harten (1985). These methods are stable, and first- or second-order accurate. The implicit TVD methods have been applied to steady-state problems. The method described by Fryxell is particularly interesting in that it can transition smoothly from an explicit to an implicit method, as can BIC-FCT. These methods are not described in detail here; we refer the interested reader to the references cited above, and to CFD textbooks. There is an extensive discussion of implicit TVD methods in Hirsch (1990).

SIMPLE

The semi-implicit method for pressure-linked equations (SIMPLE), was developed by Patankar and Spalding (1972). It is applicable to subsonic and incompressible flows and is relatively easy to apply. We recommend Tannehill et al. (1997) for an explanation somewhat more detailed than given here, and also the book by Patankar (1980) for detailed discussions. While not the most accurate method available, SIMPLE is commonly used for industrial reactive-flow problems.

SIMPLE is based on a series of predictor-corrector steps. First, define the predicted quantities, P^o and $\mathbf{v}^o$, and the correction terms, P' and $\mathbf{v}'$,

$$\begin{aligned} P &= P^o + P' \\ \mathbf{v} &= \mathbf{v}^o + \mathbf{v}'. \end{aligned} \tag{9–4.25}$$

The velocity corrections are related to P' through the momentum equation,

$$\mathbf{v}' = -A \nabla P', \tag{9–4.26}$$

where A is a time increment divided by density. Combining these two equations and substituting them into the continuity equation gives

$$\nabla^2 P' = \frac{1}{A} \nabla \cdot \mathbf{v}^o, \tag{9–4.27}$$

which is solved for P'. The quantities are updated to find new predictor values for the next iteration through finite-difference formulas such as

$$\begin{aligned} P &= P^o + P' \\ v_x &= v_x^o - \frac{A}{2\Delta x}\left(P'_{i+1,j} - P'_{i-1,j}\right) \\ v_y &= v_y^o - \frac{A}{2\Delta y}\left(P'_{i,j+1} - P'_{i,j-1}\right). \end{aligned} \tag{9–4.28}$$

The procedure is then: (1) Guess P^o on the grid; (2) Solve the momentum equations to find $\mathbf{v}_o$; (3) Solve the $\nabla \cdot P'$ equation to find P'; (4) Correct the pressure using equation (9–4.28); and (5) Replace the previous predictor values, P^o and $\mathbf{v}^o$, with the new values of P and $\mathbf{v}$, and return to step 2. The process is repeated until the solution converges.

There is often a convergence problem with this algorithm as presented above, especially when it is coupled to complex reactive flows. SIMPLER, where the R stands for *revised*, attempts to remedy this by allowing a relaxation parameter in the expression for the pressure,

$$P = P^o + \alpha_p P'. \tag{9–4.29}$$

Then $\mathbf{v}'$ is calculated the same way as in SIMPLE, and the Ps may be closer to the correct value, depending on the choice of the relaxation parameter (Patankar 1981). These methods are relatively straightforward to use, but are not the best to use for highly reactive flows or flows with steep gradients that need to be preserved.

9–5. Lagrangian Fluid Dynamics

Lagrangian fluid models approximate a continuous fluid by integrating the fluid-dynamic variables for a set of discrete nodes (or cells) that move along with the fluid. The trajectory of node i, $\mathbf{X}_i(t)$, evolves in time according to the ordinary differential equation

$$\frac{d\mathbf{X}_i(t)}{dt} = \mathbf{V}(\mathbf{X}_i, t). \tag{9–5.1}$$

Usually the fluid properties defined for a cell in an Eulerian representation, such as pressure, temperature, species densities, velocity, and momentum density, are also defined for the Lagrangian cells. As with Eulerian methods, fluid properties at locations where there are no nodes are found by interpolating from the known properties at nearby nodes. Consequently, some minimum resolution is required to adequately resolve gradients in the flow.

Sections 9–1 through 9–4 discussed a variety of algorithms in terms of how appropriate they are for describing various flow regimes. Even though these discussions generally used Eulerian representations, many of them can be implemented on any type of grid,

including Lagrangian grids. The Lagrangian algorithms described in the remainder of this chapter try to improve the solutions of the basic fluid equations by representing the fluid as Lagrangian variables moving with the local fluid velocity. These methods have met with mixed success, depending on the flow regimes being treated. In multidimensions, and even in one dimension, Lagrangian flow algorithms are intimately connected with the geometry of the problem and motion of the grid, and it is virtually impossible to separate these influences.

This section gives an overview of Lagrangian methods based on representing fluid elements. This is in contrast to those methods based on representations that track particles or vortices (discussed in Sections 9–6 and 9–7, respectively). As a prelude to describing the various types of algorithms, it is important to understand the advantages and limitations of Lagrangian methods generally (see Section 9–5.1). A specific one-dimensional method, ADINC, (see Section 9–5.2) serves as a basis for introducing a number of other issues. Generalizing one-dimensional Lagrangian methods to multidimensions is difficult, either for fixed connectivity grids (see Section 9–5.3) or for free-Lagrange methods (see Section 9–5.4) in which the moving nodes regularly exchange neighbors. The ALE algorithms, discussed throughout this chapter, where the the grid and fluid are allowed to move somewhat independently, are a possible way to avoid some of the pitfalls.

9–5.1. Advantages and Limitations of Lagrangian Methods

Lagrangian fluid dynamics shares a number of problems with the manybody dynamics approaches described in Sections 9–6 and 9–7. These methods all need to evaluate repeatedly which nodes are actually the near neighbors of any particular node. This evaluation must be done from a large set of nodes that often seem to be moving randomly. The computation of neighbors can be very expensive because each node might interact with any of $N-1$ other nodes in the system. This is called the N^2 problem, because N^2 interactions between pairs of nodes are potentially important even though each node only has a few close neighbors at any particular time. Chapter 6 suggested that the optimal Lagrangian method actually has no grid, but is only a collection of moving locations. On the other hand, some kind of localizing data structure, such as a grid, is necessary to reduce this N^2 problem to one that is more computationally manageable.

Figure 9.7 shows six different two-dimensional Lagrangian grids, each representing a different approach to dealing with a flow problem having a sharp interface between two distinct fluids. In the figure, a wave is breaking, thereby generating a region where extra resolution is needed. The figures show some adaptive grid clustering near this breaking wave.

Advantages of Lagrangian Methods

Lagrangian methods appear to offer substantial gains over Eulerian methods by reducing or eliminating the numerical diffusion caused by discrete approximations of flow across cell interfaces. In a reference frame moving at the local fluid velocity $\mathbf{v}$, the advection term $\mathbf{v} \cdot \nabla \rho$ appears only indirectly in the Lagrangian continuity equation,

$$\frac{d\rho}{dt} = \frac{\partial \rho}{\partial t} + \mathbf{v} \cdot \nabla \rho = -\rho \nabla \cdot \mathbf{v}. \tag{9–5.2}$$

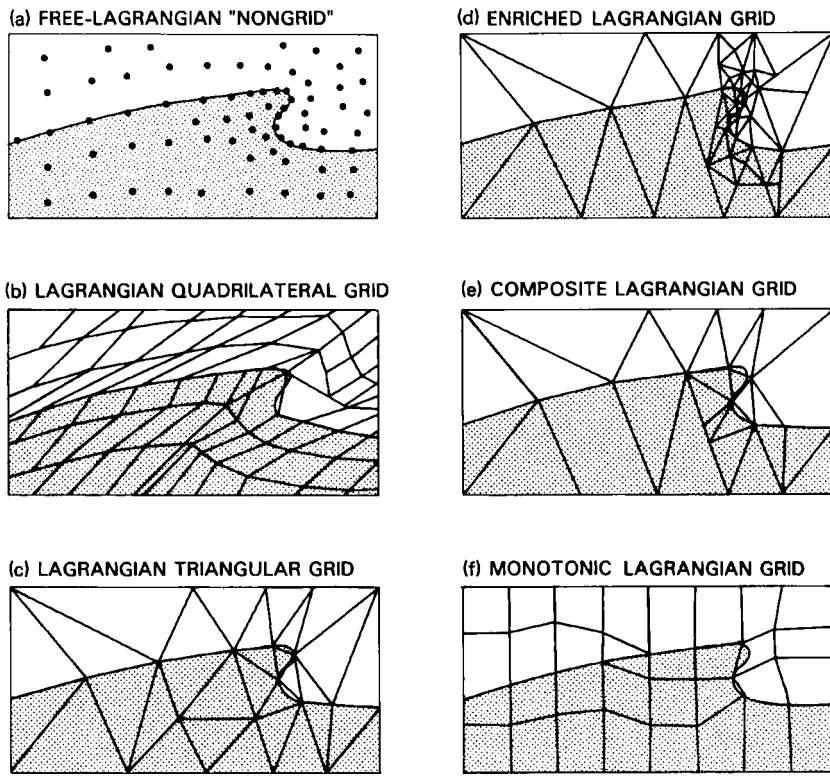

Figure 9.7. Six representative Lagrangian grids.

This troublesome advection term does not have to be differenced because it is included in the *total* or *Lagrangian time derivative* $d\rho/dt$, and so the equation appears simpler. The Lagrangian continuity equation becomes simpler still when the flow is incompressible, $\nabla \cdot \mathbf{v} = 0$, for then the right side of equation (9–5.2) disappears. Each discretized fluid element evolves through local interactions with its changing environment and with external forces. Unphysical numerical diffusion seems to disappear because there is no flow of fluid across the moving cell boundaries. Lagrangian methods also allow the user to define and track interfaces between materials by defining nodes exactly on the interface. Thus Lagrangian methods are a natural choice for treating fluid dynamics with free surfaces, separated phases, or sharp material boundaries. In addition, the paths of the fluid elements are themselves a useful flow visualization.

Practical Limitations on Lagrangian Methods

In practice, the use of Lagrangian methods in numerical simulations has generally been restricted to "well-behaved" flows. Shear, fluid separation, or even large-amplitude motions produce severe distortions of a Lagrangian grid. These distortions arise because the nodes in the fluid will move far enough that their near neighbors change in the course of a calculation. When differential operators are approximated by finite stencils on a highly distorted mesh, the approximated derivatives become inaccurate. Even when neighboring nodes stay close together, the sides of the Lagrangian cells, generally represented by straight

lines, have really become severely bent. It is the nature of complex three-dimensional flows, as discussed in Chapter 12, that small scales become distorted more rapidly than large scales. Thus fluid is constantly passing back and forth among adjacent cells, even when the nodes defining the macroscopic grid move in a Lagrangian manner. It is possible to regain some of the accuracy by interpolating the physical quantities onto a new, more regular grid, but this interpolation introduces numerical diffusion, which is just what the Lagrangian representation is trying to avoid.

Yet another intrinsic limitation arises when Lagrangian methods are applied to compressible flows. The equation for energy in an inviscid flow, equation (9–1.15), may be rewritten in a source-free conservative form by defining a new velocity field,

$$\mathbf{v}^* \equiv \left(\frac{E+P}{E}\right)\mathbf{v}, \tag{9–5.3}$$

which is always parallel to $\mathbf{v}$. Equation (9–1.15) then becomes a simple continuity equation for the energy density,

$$\frac{\partial E}{\partial t} = -\nabla \cdot E\,\mathbf{v}^*. \tag{9–5.4}$$

This has the same form as the mass conservation equation (9–1.13), but the velocities are different whenever the pressure is greater than zero. Thus a *grid moving at velocity* $\mathbf{v}$ *is Lagrangian for density but not for energy. Conversely, a grid moving at* $\mathbf{v}^*$ *is Lagrangian for energy but not for density*. Energy and momentum still move through the interfaces of a Lagrangian grid much as they would if the grid were Eulerian and stationary. Therefore, Lagrangian fluid representations still suffer numerical diffusion. In compressible reactive flows, the full set of equations is usually solved simultaneously, so it is not generally possible to avoid the Eulerian aspects of the problem.

Multidimensional Lagrangian algorithms are not easy to program or use. The logic of grid restructuring (or remapping) is essential and quite complicated. Many restructuring algorithms that seem "obvious" have required time-consuming efforts to find a usable implementation.

9–5.2. A One-Dimensional Implicit Lagrangian Algorithm

ADINC (which stands for ADiabatic and INCompressible [Boris 1979]), is an implicit one-dimensional Lagrangian algorithm designed to solve the mass, momentum, and energy equations for systems where the fluid velocity is small compared to the sound speed. It is thus an algorithm for the slow-flow regime, for which Eulerian representations were described in Section 9–4. ADINC has been used as the flow solver for several very different kinds of reactive-flow applications, ranging from low-velocity combustion problems (Kailasanath 1991) to magnetohydrodynamics and controlled thermonuclear fusion (Cooper et al. 1980). Nonadiabatic processes, such as energy release, were built into the reactive-flows models by the timestep-splitting methods discussed in Chapter 11. The development of the combustion model, FLAME1D, was originally motivated by the need to have an accurate convection algorithm without numerical diffusion for treating the time-dependent behavior of one-dimensional flames with minimal computational resolution.

There can be rapid local energy release in gas-phase flame problems, but the fluid velocities resulting from the expansion are much slower than the sound speed. In these cases, it is usually better to treat the sound waves implicitly. In other systems, rather sharp interfaces between regions with different compositions or different temperatures must be maintained, yet the fluids interact across the interface. This could benefit from a Lagrangian description so that the exact interface location is available at any time. ADINC deals with these two problems directly. It also accepts a general equation of state and complicated boundary conditions. Nonideal effects, such as energy release and physical diffusion processes, are added by timestep splitting.

The Basic Equations

The density, energy, and momentum conservation equations can be rewritten in Lagrangian form to give

$$\frac{d\rho}{dt} = -\rho \nabla \cdot \mathbf{v}, \tag{9–5.5}$$

$$\frac{ds}{dt} = 0, \tag{9–5.6}$$

$$\rho \frac{d\mathbf{v}}{dt} = -\nabla P. \tag{9–5.7}$$

Here the energy equation has been replaced by an equation for the entropy $s(r)$, where r is a general one-dimensional space variable. The entropy is constant in each fluid element bounded by Lagrangian interfaces. This removes numerical diffusion from the energy continuity equation, but makes exact conservation of energy difficult to achieve. For example, viscous heating for shocks cannot be well represented in this model. This is is not a problem, however, because we are concentrating on the slow-flow regime.

This set of equations may be closed by the equation of state

$$\rho(P, s, \ldots) = \rho_c + \left(\frac{P}{s}\right)^{1/\gamma}. \tag{9–5.8}$$

When $\rho_c = 0$, equation (9–5.8) represents adiabatic compression and expansion in an ideal gas, where $1.2 \leq \gamma \leq 1.7$. In reactive flows, γ is a function of temperature, density, and the species mass fractions. When $\rho_c \neq 0$, this equation represents a mildly compressible liquid. For water, $\rho_c = 1$ g/cm^3 and $s = 2.5 \times 10^{11}$cm/s^2. In this model, a pressure of 250 Kbar, or 2.5×10^{11} dynes/cm^2, causes a substantial compression in the water. More complicated equations of state for solids, liquids, or plasmas may replace equation (9–5.8).

The equation of state appears in the form $\rho(P, s, \ldots)$ for two reasons. First, the density is a much less sensitive function of the pressure for liquids and solids than the pressure is of the density. During the iteration process, finite errors in pressure and density are expected. In the alternate form $P(\rho, s, \ldots)$, the errors in density result in large fluctuations in the pressure. For gases and plasmas, the two forms of the equations of state have about the same accuracy, but the $\rho(P, s, \ldots)$ formulation is still preferred for atmospheric weather models. The second reason is that the algorithm can be used for tracking material interfaces and contact surfaces. At such an interface, the pressure is continuous but the density does not

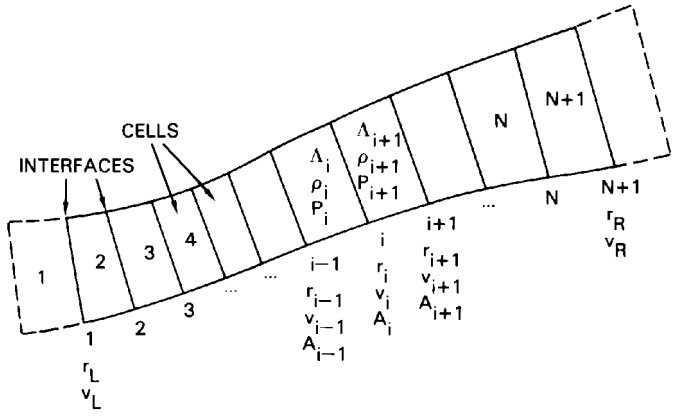

Figure 9.8. A general one-dimensional region of variable cross-sectional area $A(r)$ as used in the Lagrangian algorithm ADINC.

have to be. Therefore, finite differences in the pressure are more accurate computationally than the mathematically equivalent differences in the density.

During a timestep, ρ_c, γ, and s are treated as constants and only the variation of ρ with P is considered. ADINC calculates ρ, given an approximation to P that does not involve the temperature. The density from the equation of state is compared to the density derived from equation (9–5.5), and the difference is forced to zero by adjusting the locations of the cell interfaces using a quadratically convergent iteration algorithm. ADINC iterates equations (9–5.5), (9–5.6), and (9–5.7) and produces an improved pressure approximation. The iteration uses the derivative $d\Lambda/dP$

$$\frac{1}{\Lambda}\frac{d\Lambda}{dP} = -\frac{1}{\gamma\rho P}\left(\frac{P}{s}\right)^{\frac{1}{\gamma}}, \tag{9–5.9}$$

where Λ is the volume of a computational cell. To complete the set of equations, the geometry of the calculation must be given. ADINC incorporates four one-dimensional geometries: Cartesian, cylindrical, spherical, and power-series, and variable cross-section "nozzle" coordinates. Figure 9.8 shows a general one-dimensional domain of variable cross-sectional area $A(r)$.

The Numerical Algorithm

A typical computational grid for ADINC, Figure 9.8, consists of N cells of volume $\{\Lambda_i\}$, $i = 2, 3, \ldots, N+1$. The cells are bounded by $N+1$ interfaces of area $\{A_i\}$ located at $\{r_i\}$, $i = 1, \ldots, N+1$. Thus $A_i \equiv A(r_i)$, where $A(r)$ is determined by the chosen geometry. The cell volumes, Λ_i are given by

$$\Lambda_i = \int_{r_{i-1}}^{r_i} A(r')\,dr'. \tag{9–5.10}$$

The locations of the cell centers are defined by

$$R_i \equiv \frac{A_i r_{i-1} + A_{i-1} r_i}{A_i + A_{i-1}}, \tag{9–5.11}$$

and R_i always lies between r_{i-1} and r_i.

The cell-interface positions $\{r_i\}$ change in time according to equation (9–5.1), rewritten here as

$$\frac{dr_i}{dt} \equiv v_i. \tag{9–5.12}$$

This is discretized to

$$r_i^n = r_i^o + \Delta t\left[\epsilon_r v_i^o + (1-\epsilon_r)v_i^n\right]. \tag{9–5.13}$$

The superscript n indicates variables at the new time, $t+\Delta t$, and the superscript o indicates variables at the old time, t. The quantity ϵ_r is the explicitness parameter for the interface positions, $0 \le \epsilon_r \le 1$. When $\epsilon_r = \frac{1}{2}$, the method is centered and nominally most accurate. To take long timesteps, we must use $\epsilon_r \le \frac{1}{2}$ to keep the algorithm stable. When $\epsilon_r = 0$, the calculation is fully implicit. This is the most stable choice, but it is only first-order accurate. It is possible to vary ϵ_r from cell to cell and from timestep to timestep.

The velocity is evaluated at cell interfaces from equation (9–5.7) by

$$\frac{dv_i}{dt} \equiv \frac{-1}{\rho_{\text{interface } i}} \left.\frac{\partial P}{\partial r}\right|_{\text{interface } i}, \tag{9–5.14}$$

where the density and pressure gradient must be estimated at cell interfaces. The discretization used for equation (9–5.14) is

$$v_i^n = v_i^o - \frac{\Delta t\ \epsilon_v}{\langle \rho \Delta r\rangle_{i+\frac{1}{2}}}\left(P_{i+1}^o - P_i^o\right) - \frac{\Delta t(1-\epsilon_v)}{\langle \rho \Delta r\rangle_{i+\frac{1}{2}}}\left(P_{i+1}^n - P_i^n\right). \tag{9–5.15}$$

Here ϵ_v is the explicitness parameter for the interface velocity and has the same properties as ϵ_r. Although ϵ_v and ϵ_r can be distinct, we generally use the same value.

The interface averaged mass, $\langle \rho \Delta r\rangle_{i+\frac{1}{2}}$, is both a spatial and a temporal average. The average is defined so that the discretization in equation (9–5.15) is insensitive to numerical errors arising from large density discontinuities that can occur at material interfaces. Figure 9.9 shows the two cells i and $i+1$ that straddle interface i. The pressures P_i and P_{i+1} are defined at cell centers R_i and R_{i+1} as shown, and the densities ρ_i and ρ_{i+1} are assumed constant throughout their respective cells. Because ρ_i and ρ_{i+1} may differ significantly, using a straight line to model the pressure gradient from R_i to R_{i+1} would mean that the local acceleration is different for the fluid just to the right and to the left of interface i.

If the fluid were allowed to move according to these different accelerations, the fluids would either overlap or separate near the interface i after a short time. To prevent this, a suitably weighted average pressure P_i^* is calculated for interface i. This interface pressure is chosen so that the fluid acceleration on the left of the interface equals the acceleration calculated in the fluid on the right. Then

$$P_i^* \equiv \frac{P_{i+1}\rho_i(r_i - R_i) + P_i\rho_{i+1}(R_{i+1} - r_i)}{\rho_i(r_i - R_i) + \rho_{i+1}(R_{i+1} - r_i)}. \tag{9–5.16}$$

Using equation (9–5.16), the average for $\langle \rho \Delta r\rangle_{i+\frac{1}{2}}$ can be chosen to eliminate these interface pressures $\{P_i^*\}$ from further consideration. The spatial part of the average that

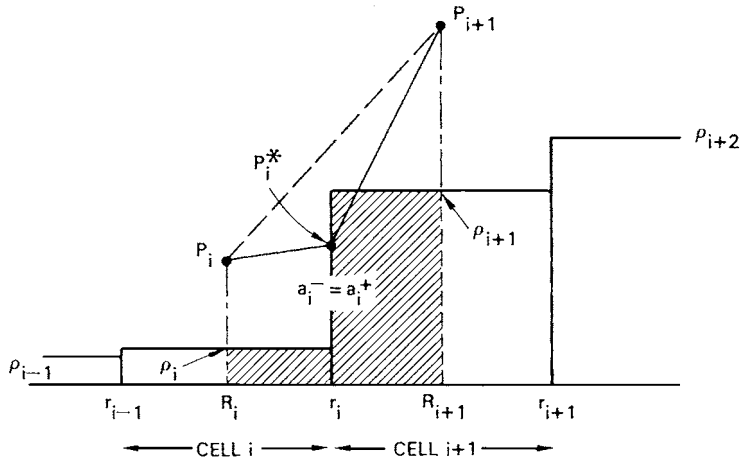

Figure 9.9. The interfaces between the two cells, i and $i+1$, that straddle interface i. The diagram schematically shows the interface-acceleration algorithm used in the one-dimensional algorithm ADINC.

accomplishes this *acceleration matching* is

$$\langle \rho \Delta r \rangle_{i+\frac{1}{2}} \equiv \rho_{i+1}(R_{i+1} - r_i) + \rho_i(r_i - R_i). \tag{9–5.17}$$

Acceleration matching can be very useful, even in Eulerian models.

The temporal average in ADINC for equation (9–5.17) is chosen as

$$\langle \rho \Delta r \rangle_{i+\frac{1}{2}} \equiv \left[\rho^p_{i+1}\left(R^h_{i+1} - r^h_i\right) + \rho^p_i\left(r^h_i - R^h_i\right)\right], \tag{9–5.18}$$

where the superscript p (for "previous") indicates the most recently iterated approximation to the new value of the variable, in this case ρ^n_i. The superscript h is used to indicate the "half-time" average. In equation (9–5.18),

$$r^h_i \equiv \frac{1}{2}\left(r^o_i + r^p_i\right) \quad \text{and} \quad R^h_i \equiv \frac{1}{2}\left(R^o_i + R^p_i\right). \tag{9–5.19}$$

These are not the only possible averages that could be chosen, but no problems arising from these particular choices have been identified.

We now derive the tridiagonal (implicit) equation for P^n_i. Equation (9–5.15) for the momentum can be simplified to

$$v^n_i = \alpha_i - \beta_i\left(P^n_{i+1} - P^n_i\right), \qquad i = 1, \ldots, N, \tag{9–5.20}$$

by using these auxiliary definitions,

$$\begin{aligned} \alpha_i &\equiv v^o_i - \frac{\Delta t\ \epsilon_v}{\langle \rho \Delta r \rangle_{i+\frac{1}{2}}}\left(P^o_{i+1} - P^o_i\right), \\ \beta_i &\equiv \frac{\Delta t(1-\epsilon_v)}{\langle \rho \Delta r \rangle_{i+\frac{1}{2}}}. \end{aligned} \tag{9–5.21}$$

Now we need to enforce that $\Lambda_i^{n(\text{eos})}$ is equal to $\Lambda_i^{n(\text{fd})}$, where $\Lambda_i^{n(\text{eos})}$ is the new cell volume computed from the equation of state using the new value of the pressure, and $\Lambda_i^{n(\text{fd})}$ is the new cell volume computed from the fluid dynamics. The quantity $\Lambda_i^{n(\text{fd})}$ is based on how much the new pressure values move the cell interfaces. At successive iterations, indexed by p, the difference

$$\Delta\Lambda_i^p \equiv \Lambda_i^{p(\text{eos})}\big(P_i^p, s_i, \dots\big) - \Lambda_i^{p(\text{fd})}\big(\{r_i^p\}\big), \tag{9–5.22}$$

must approach zero. Changing P_i^p to P_i^n varies both terms in equation (9–5.22). The quantity r_i^p converges to r_i^n as a function of the pressure through equation (9–5.20) and equation (9–5.13). The Newton-Raphson algorithm gives a quadratically convergent iteration to obtain the desired solution at time $t + \Delta t$,

$$\Lambda_i^{n(\text{eos})}\big(P_i^n, s_i, \dots\big) = \Lambda_i^{n(\text{fd})}\big(\{r_i^n\}\big). \tag{9–5.23}$$

The difference between $\Lambda_i^{p(\text{fd})}$, which is known at each iteration, and the desired $\Lambda_i^{n(\text{fd})} = \Lambda_i^{n(\text{eos})} = \Lambda_i^n$ can be written in terms of the cell interface areas and the desired new fluid velocities of the Lagrangian cell interfaces,

$$\Lambda_i^n - \Lambda_i^{p(\text{fd})} \approx (1 - \epsilon_r)\Delta t\big[A_i^h\big(v_i^n - v_i^p\big) - A_{i-1}^h\big(v_{i-1}^n - v_{i-1}^p\big)\big]. \tag{9–5.24}$$

The same treatment of the equation of state gives

$$\Lambda_i^n - \Lambda_i^{p(\text{eos})} \approx \big(P_i^n - P_i^p\big)\frac{\partial\Lambda}{\partial P}\bigg|^{p(\text{eos})}. \tag{9–5.25}$$

By defining

$$d_i^+ \equiv -(1 - \epsilon_r)\Delta t\ A_i^h, \qquad d_i^- \equiv -(1 - \epsilon_r)\Delta t\ A_{i-1}^h, \tag{9–5.26}$$

equation (9–5.24) becomes

$$\Lambda_i^n - \Lambda^{p(\text{fd})} \approx -d_i^+\big(v_i^n - v_i^p\big) + d_i^-\big(v_{i-1}^n - v_{i-1}^p\big). \tag{9–5.27}$$

Equating Λ_i^n in equations (9–5.25) and (9–5.27) gives

$$\Delta\Lambda_i^p + \big(P_i^n - P_i^p\big)\frac{\partial\Lambda}{\partial P}\bigg|_i^{p(\text{eos})} \approx -d_i^+\big(v_i^n - v_i^p\big) + d_i^-\big(v_{i-1}^n - v_{i-1}^p\big). \tag{9–5.28}$$

Now, collecting the known quantities

$$c_i \equiv P_i^p\frac{\partial\Lambda}{\partial P}\bigg|_i^{p(\text{eos})} + d_i^+ v_i^p - d_i^- v_{i-1}^p - \Delta\Lambda_i^p, \tag{9–5.29}$$

equation (9–5.28) becomes

$$P_i^n\frac{\partial\Lambda}{\partial P}\bigg|_i^{p(\text{eos})} + d_i^+ v_i^n - d_i^- v_{i-1}^n \approx c_i. \tag{9–5.30}$$

This is the implicit tridiagonal, linear equation that must be solved for the estimated new pressures $\{P_i^n\}$. Equation (9–5.20) is used to eliminate $\{v_i^n\}$ in terms of $\{P_i^n\}$. Expanding

equation (9–5.30) gives

$$P_i^n \frac{\partial \Lambda}{\partial P}\bigg|_i^{p(\text{eos})} + d_i^+\left[\alpha_i - \beta_i\left(P_{i+1}^n - P_i^n\right)\right] - d_i^-\left[\alpha_{i-1} - \beta_{i-1}\left(P_i^n - P_{i-1}^n\right)\right] \approx c_i. \tag{9–5.31}$$

Equation (9–5.31) is the basic tridiagonal equation solved by ADINC. Iteration is necessary because equations (9–5.25) and (9–5.27) are only approximate equalities. The iteration is quadratically convergent because only second-order terms are neglected at each stage of the iteration.

When integrating the fluid-dynamic equations, this algorithm assumes that each interface moves in a fully Lagrangian manner according to equation (9–5.12). The change in density from one timestep to the next in a cell is therefore given simply by the change in cell volume using mass conservation,

$$\rho_i^n \Lambda_i^n \;\equiv\; \Delta M_i \;\equiv\; \rho_i^o \Lambda_i^o. \tag{9–5.32}$$

When individual species number densities must be advanced, they are also found using equation (9-5.32).

Some General Comments

The interface positions $\{r_i\}$ must increase monotonically with increasing i, and the interface areas $\{A_i\}$ must all be positive. Immediate problems occur when cell interfaces cross each other. This can occur for large timesteps even though the implicit algorithm given above is nominally stable for sound waves at arbitrary timesteps. Such crossings in one dimension forebode the more complicated grid distortions and "hourglass" motions that occur in multidimensions (see, for example, Caramana and Shashkov [1998]).

To prevent interface crossings in one dimension, a Courant condition must still be satisfied for the flow velocities $\{v_i\}$, even though the Mach number is low throughout the gas. If $\{v_i^n\}$ becomes large enough, $\{r_i^n\}$ can cross adjacent interfaces even though the original timestep estimate should prevent this. The maximum timestep is limited by

$$\Delta t \le \frac{1}{2} \min\left\{\left(\frac{r_{i+1} - r_i}{|v_i| + \delta}\right), \left(\frac{r_i - r_{i-1}}{|v_i| + \delta}\right)\right\}, \qquad i = 2, \ldots, N. \tag{9–5.33}$$

A very small number, δ, is added to $|v_i|$ to prevent dividing by zero in equation (9–5.33). This equation is a conservative estimate and usually longer timesteps may be taken.

The cell-interface positions are advanced using an average of the new and old velocities in equation (9–5.13). Because the timestep has to be estimated at the beginning of a cycle, it is possible for $\{v_i^o\}$ to be small or zero while $\{v_i^n\}$ can be large. A user-supplied maximum timestep usually provides a suitable limit.

Using the algorithm given above for estimating the timestep, ADINC subcycles the fluid dynamics as often as necessary to integrate over the time interval specified. Within each timestep (or subcycle), ADINC performs a convergence calculation on the new iterated tridiagonal pressure solution. The convergence condition used for each cell is

$$\frac{|P_i^n - P_i^p|}{P_{\max}} < 10^{-9}\sqrt{\frac{M_{\max}}{M_{\min}}}, \tag{9–5.34}$$

where P_{max} is the largest pressure in the system, M_{max} is the largest cell mass in the system, and M_{min} is the smallest cell mass in the system. Equation (9–5.34) is heuristic, but works well. Using the timestep condition given above, convergence in $\{P_i^n\}$ is often obtained in two iterations and almost always in three or four. The maximum number of interactions is limited to six, and this generally gives high accuracy.

Because ADINC is Lagrangian, it should be more accurate than the Eulerian BIC-FCT algorithm described earlier. BIC-FCT, however, can also handle shocks with short timesteps, conserves energy, and can also be generalized easily to multidimensions. It is unfortunate that equally robust and efficient multidimensional generalization of ADINC have not been developed, and why this is the case is explained next. Instead, numerical developments have focused on two different approaches, arbitrary Lagrange-Eulerian (ALE) algorithms that are often implemented on unstructured finite-element grids, and dynamic adaptive mesh refinement algorithms that are usually applied on Eulerian grids.

9–5.3. Multidimensional Lagrangian Algorithms

Extending the one-dimensional Lagrangian model described in the previous section to multidimensional flows is only straightforward when the grid does not become highly distorted as the flow evolves. This occurs in two circumstances: (1) when the duration of the computation is short so that the motions of the nodes are very limited, or (2) when the underlying motion is primarily a potential flow so that neighboring nodes remain neighbors to each other. A number of multidimensional Lagrangian approaches have been explored, including the ALE algorithms, free-Lagrange methods, smooth-particle hydrodynamics (SPH), quasiparticle and lattice-gas (LG) methods, and vortex-dynamics methods. We note that adaptive mesh refinement with intrinsically Eulerian algorithms (see Chapter 6), is emerging as the most useful and thus the preferred approach for reactive flows.

Lagrangian representations use quadrilateral, triangular, and mixed meshes in two dimensions, as illustrated in Figure 9.7. These representations generalize to hexahedral, tetrahedral, and mixed meshes in three dimensions. Voronoi meshes, which consist of polygons formed by connecting the centroids of triangles, are commonly used in both two and three dimensions. There are other more nearly mesh-free methods, the free-Lagrange methods, in which there is still a grid, but the grid connectivity is changed "freely," as needed. In any of these Lagrangian methods, cell interfaces are rarely orthogonal. Thus many of the finite-difference operators we would like to use become less accurate and more expensive.

Figures 9.7b and 9.7c show Lagrangian grids composed of quadrilaterals and triangles, respectively. The distortions, which usually accompany rotational flow in the quadrilateral grid, are shown where the interface is becoming tangled near the breaking wave. The advantage of the triangular grid is that the nodes can be reconnected to produce a better grid whenever the triangles become too elongated or compressed. The grid tangling and "bowties" that occur with quadrilateral Lagrangian grids are absent. Figure 9.7d shows a Lagrangian triangular-grid representation that has been refined locally by dividing cells. The price for this finely focused, high resolution is the need to work with algorithms that are not easily optimized for use on parallel processing or multiprocessing computers. Their relative computational inefficiency is generally compensated by the small number of zones needed for excellent resolution.

Depending on a number of factors – including the flow conditions, the length of time over which the computation must be performed, and the flow geometry – one or more of these problems may become very important. For each particular physical problem, some of these Lagrangian approaches may be more suitable than others. Because of the intrinsic problems with grid distortion, there are as yet no simple, generally accepted, robust Lagrangian methods, even for simple finite-difference and finite-volume algorithms.

There are only two options in a Lagrangian simulation when the grid is becoming unacceptably distorted. Either the grid simply must stop distorting, or the grid has to be continually restructured to remove the distortions. If the flow is still evolving and the grid distortion is halted, numerical diffusion reappears because the fluid now passes through the grid interfaces. This is the case for the ALE algorithms considered below. If the distortion is significant, the errors can be larger than would occur by using an Eulerian grid throughout the calculation. Restructuring the grid results in the free-Lagrange methods (see Section 9–5.4). Nonetheless, reconnecting the grid requires interpolation, which itself is a form of numerical diffusion.

Arbitrary Lagrangian-Eulerian Algorithms

Given the freedom to move the grid, there is no reason why the grid motion *must* follow the fluid particles. Once the computational cost of updating the mesh geometry and computing on irregular cells is paid, it is possible to consider more general prescriptions for moving the grid that might overcome, or at least minimize, the shortcomings of using a Lagrangian representation. This class of algorithms has become known as *ALE* (Hirt 1971; Hirt, Amsden, and Cook 1974).

ALE relaxes the requirements of a fully Lagrangian representation when the quadrilateral grid became highly tangled due to intense local vorticity. For example, one ALE approach to restricting grid movement is to define a quantitative criterion for halting the grid distortion, by somehow removing the motions that cause the distortion. This allows fluid variables to convect across the cell interfaces at a rate determined by the difference between the local fluid velocity and the permitted motion of the cell interfaces. Some numerical diffusion is introduced by this process.

ALE algorithms have evolved to incorporate Lagrangian motion primarily normal to material interfaces, where it is important to keep different materials separated. Today ALE representations are the preferred approach for unstructured grid models with moving cell boundaries (see, for example, Baum, Luo, and Löhner [1994], Ramamurti, Löhner, and Sandberg [1994], and Löhner and Yang [1996]).

9–5.4. Free-Lagrange Methods

In the purest sense, a *free-Lagrange* method has no grid, as shown schematically in Figure 9.7a. No particular grid or mesh is constructed to connect the nodes. Fluid derivatives are evaluated directly from Lagrangian node quantities. Vortex dynamics and boundary-integral formulations fall into this general category. Smooth-particle hydrodynamics (see Section 9–6.2) is perhaps the purest of free-Lagrange methods. The price for removing the grid is the need to compute all of the node interactions by pairs instead of computing just the interactions of nearby nodes.

In practice, the term *free Lagrange* is also applied to Lagrangian methods with grids if the grids can vary their connectivity in the course of a calculation. Such methods were specifically developed for computing flows in which motion near material interfaces unacceptably distorts a fixed-connectivity grid. This type of distortion arises, for example, in computations of material deformations due to impact, free-surface (air-water) hydrodynamics, shear layers in reacting jets, and multiphase fluids in which the shape and dynamics of the interface are important. In fact, fluid-dynamics algorithms based on free-Lagrange methods (Fritts, Crowley, and Trease 1985; Trease, Fritts, and Crowley 1991) generally use a grid, so that local derivatives may be calculated accurately. To be efficient, these algorithms depend on the fact that only a small fraction of the grid connectivity must be changed at each timestep. Because these methods focus particularly on dynamically changing interfaces, they are often used for problems in which the physics normal to an evolving interface should be resolved at a very different scale from the physics along the interface.

General-Connectivity Lagrangian Grids

Free-Lagrange algorithms based on *general-connectivity* triangular or tetrahedral grids (Figure 9.7c) allow the nodes to change their neighbors (i.e., reconnect) continually during the course of a calculation. The node reconnections are chosen to maintain as regular a grid as possible, a property that maintains accuracy in derivative evaluations. Fritts and Boris (1979) developed one of the earliest of these algorithms. Their method uses a grid of triangles (or tetrahedra) and an implicit pressure correction to describe incompressible, multiphase flows. This method was extended to three dimensions by Fritts (1985), and the issues of incorporating surface tension at material surfaces were investigated by Fyfe, Oran, and Fritts (1988).

Other implementations of general-connectivity grids include a similar formulation by Crowley (1985), Börgers and Peskin (1985), the explicit X3D model (Mandell and Trease 1989; Cerutti and Trease 1991), and TRIX (Gittings 1991) that uses a wide variety of regridding methods and Godunov computations to find the stress and velocities at the interfaces between cells. Eltgroth (1985, 1991) described a free-Lagrange method that allowed independent timesteps in different regions of the mesh, as governed by local physics.

The conference volumes edited by Fritts et al. (1985) and later by Trease et al. (1991) contain articles that summarize different free-Lagrange algorithms and their various applications. An enormous amount of work was devoted in the 1980's to developing free-Lagrange general-connectivity methods, which were viewed then as a promising way to eliminate numerical diffusion and produce accurate solutions at material boundaries. These intense efforts were subsequently reduced or phased out, primarily because several fundamental problems were not solved. First, these methods are expensive and complex compared to more standard Eulerian approaches. Second, numerical instabilities arose when the methods were used, and especially when they were extended to reactive flows. Today, these approaches have evolved into the unstructured grid methods and the ALE approaches described previously.

The Monotonic Lagrangian Grid

The distinction between methods with grids and those without is not a strong one. In many respects, the differences can degenerate to semantic arguments. To claim any degree of computational efficiency, even grid-free methods need some type of data structure and

algorithms to find near neighbors quickly for evaluating spatial derivatives. These auxiliary data structures, in effect, define a grid. Even tree structures, when viewed appropriately, define a mesh.

The monotonic Lagrangian grid (MLG) is just such an ambiguous free-Lagrangian data structure for storing the positions and other data needed to describe N moving nodes (Boris 1986; Lambrakos and Boris 1987). A node with three spatial coordinates has three indices in the MLG data arrays. The data relating to each node are stored in the memory locations indicated by these indices. Thus nodes that are close to each other in real space are always near neighbors in the MLG data arrays. A logically regular grid has been constructed with a simple mapping between node indices and positions. This simplifies and speeds most calculations of the interactions between nodes. In three dimensions, the derivatives can be evaluated as on a general hexahedral grid.

A computer program based on the MLG data structure does not need to check $N-1$ possible distances to find which nodes are close to a particular node. The indices of the neighboring nodes are automatically known because the MLG node indices vary monotonically in all directions with the Lagrangian node coordinates. The cost of the algorithm in practical calculations is dominated by the calculation of the interactions of nodes with their near neighbors, and the timing thus scales as N. Figure 9.10 shows an example of a small two-dimensional MLG. Because spatial coordinates define a natural ordering of positions, it is always possible to associate two grid indices with a set of random locations in a two-dimensional space (as described in detail in the references). The two-dimensional MLG shown in the figure satisfies the following indexing constraints:

$$x(i, j) \le x(i+1, j) \qquad i = 1, \ldots, N-1; \qquad j = 1, \ldots, N \tag{9–5.35}$$

$$y(i, j) \le y(i, j+1) \qquad i = 1, \ldots, N; \qquad j = 1, \ldots, N-1. \tag{9–5.36}$$

The nodes are shown at their irregular spatial locations, but they are indexed regularly in the MLG by a monotonic mapping between the grid indices and the locations. Each grid line in each spatial direction is forced to be a monotone index mapping.

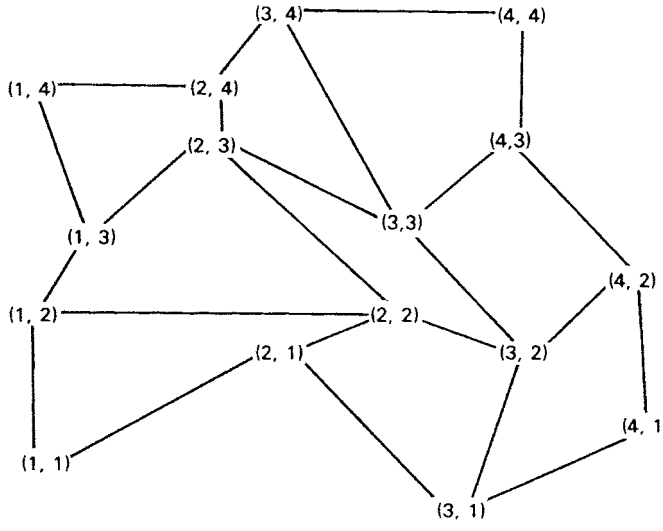

Figure 9.10. An example of a two-dimensional monotonic Lagrangian grid.

A construction algorithm that scales as $N \log N$ can be used to build an MLG from randomly located nodes (Boris 1986). Thus, an MLG can always be found for arbitrary data although it is not necessarily unique. When node motions in physical space destroy some of the monotonicity conditions, another fast $N \log N$ algorithm exists to exchange the node data between adjacent cells until MLG order is restored. Node data stored in the MLG according to this prescription are at most two or three nodes away from the neighboring nodes that can affect them. Therefore, for near-neighbor gradient, derivative, or force calculations, only a fraction of the possible node-node interactions need to be considered. One advantage of such a "grid" is that computations can be vectorized and parallelized efficiently because nearby nodes are indexed in contiguous computer memory.

The MLG data structure combines some of the features of both Lagrangian and Eulerian representations. Some nodes can be located at interfaces and move with them, so these interfaces may be resolved in a Lagrangian manner without numerical diffusion. The MLG also has the simple global structure and regular indexing of Eulerian grids, so the efficiency of the method can approach the efficiency of fixed-grid Eulerian algorithms. The local structure of an MLG is not quite as good as with Eulerian or ALE grids. Typically a factor of two or more nodes is needed to obtain the same accuracy as locally adapting Lagrangian triangular grids. The potential gain with the MLG comes from parallel computing and the ease with which it can be load balanced. To date, the primary use of the MLG has been to represent the movement and interactions of particles such as atoms or molecules, droplets, or dust particles, and not to represent fluid elements. Some of these representations are discussed in the next section.

Other Free-Lagrange Options

Free-Lagrangian approaches that use grids and maintain general connectivity to achieve good quality grids are expensive. When implemented on parallel computers, they generally require a large amount of interprocessor communications. It is also possible to develop methods without a grid in the usual sense, in which the fluid variables are defined at the Lagrangian nodes, but their values and the gradients between the nodes are determined by a weighting procedure. All of these approaches promise the advantages and appear to suffer, in one manner or another, the general drawbacks discussed in Section 9–5.1.

Tree data structures and their associated fast algorithms are a good alternative for organizing free-Lagrange and manybody data to avoid the N^2 problem. In Chapter 6 we discussed trees in connection with unstructured grid generation and AMR, but tree data structures were originally developed for molecular dynamics and manybody dynamics computations. Recent developments (Khokhlov 1998) have minimized both the numerical operations required to maintain the tree structure and the computer memory requirements in a framework that lends itself to efficient, massively parallel processing.

9–6. Lagrangian Particle and Quasiparticle Methods

Particle representations are another approach to solving fluid-dynamics problems. From the earliest days of CFD, it was understood that the major aspects of fluid dynamics could be reproduced by tracking a large number of particles to represent the fluid. An introductory discussion of the numerical representations for Lagrangian particle and macroparticle

methods is given in Chapter 6. Here we review the material in Sections 6–1 and 6–2.5, and add to those discussions.

Solving the equations of molecular dynamics for all the atoms and molecules of a gas or liquid would, in principle, describe the fluid dynamics if we had a large and fast enough computer. This is not practical, and the actual number of particles used in a simulation is typically many orders of magnitude less than in the real problem. In this sense, particle models are numerically "grainy." For example, it is typical to simulate a galaxy, consisting of 10^{11} stars with 10^5 simulation particles, each with the effective mass of about 10^6 stars. For fluids, 1 cm^3 of water, with $\sim 10^{22}$ molecules, is simulated with at most 10^6 simulation particles. This graininess leads to a number of numerical problems, some of which are discussed by Birdsall and Langdon (1985) for plasma simulations and by Bird (1994) for fluid simulations. The term *macroparticle* was coined to remind us that the simulation particles are much bigger than the actual particles comprising the fluid.

There is a difference between particle models and quasiparticle models. In particle models, the Lagrangian macroparticles freely interpenetrate, so several can occupy the same space. A velocity distribution is then a valid concept. Quasiparticle models use macroparticles to represent blobs of fluid moving with the flow. These quasiparticles have dimensions comparable to the separation between neighboring quasiparticles and their velocity is meant to reflect the local fluid velocity. Sometimes this distinction may be somewhat artificial, as is the distinction between quasiparticle and free-Lagrange approaches. The important point is that particle methods can only recover smooth fluid behavior by statistically averaging over a large number of macroparticles, whereas quasiparticle models attempt to simulate continuous fluid-like behavior by constructing the representation in a way that does not rely so heavily on statistical averages.

When a number of macroparticles can occupy essentially the same location in space, and yet move with significantly different velocities, the model can be used to describe collisionless systems, such as plasmas on the microscopic scale or stars in self-gravitating systems. The original algorithms used for collisionless flows stem from work by Hockney (1965, 1966) and Buneman (1967). Collisionless macroparticle methods are reviewed by Birdsall and Fuss (1969), Langdon and Lasinski (1976), and Birdsall and Langdon (1985).

In this book, microscopic particle representations, such as molecular dynamics or plasma simulation are not considered. The material in this section concerns the use of particle methods as alternate approaches to CFD. These include many of the quasiparticle methods, smooth-particle hydrodynamics, and lattice-gas automata. The techniques and data structures introduced in Section 9–5 to manage and manipulate the nodes for multidimensional Lagrangian methods are also applicable to particle and quasiparticle methods. Although many macroparticle models do not need a grid, an efficient solution of the near-neighbors problem is needed to compute the collisions or macroscopic forces between them. As a further confusion in attempts to classify methods, some particle methods use a grid to construct and even smooth the continuous variables that are needed for fluid dynamics.

9–6.1. Quasiparticle Methods

Quasiparticle methods solve convection problems by simulating interactions between the Lagrangian, particle-like blobs that make up the fluid. The collective, continuum behavior

of the fluid is extracted as suitable averages over the quasiparticle locations and their accumulated properties. If a large number N of quasiparticles are initialized at positions $\{\mathbf{x}_j(0)\},\ j = 1, \ldots N$ at time $t = 0$, their later positions can be found at time t by integrating the (Lagrangian) trajectory equations

$$\frac{d\mathbf{x}_j}{dt} = \mathbf{v}_j(t), \tag{9–6.1}$$

where the velocity $\mathbf{v}_j$ of each quasiparticle j is found by integrating a force law

$$\frac{d\mathbf{v}_j}{dt} = \mathbf{f}_j(\{\mathbf{x}_k\}, \{\mathbf{v}_k\}, t). \tag{9–6.2}$$

Here the force $\mathbf{f}_j$ can depend on the positions and velocities of the other particles and their pressure. The fluid mass density is found by summing the masses m_j of all the particles j at or near the ith location,

$$\rho(x_i) = \left\langle \sum_i m_i\, \delta(\mathbf{x}_j - \mathbf{x}_i) \right\rangle. \tag{9–6.3}$$

The fluid momentum density is found by summing the particle momenta,

$$\rho\mathbf{v}(x_i) = \left\langle \sum_j m_j \mathbf{v}_j \delta(\mathbf{x}_j - \mathbf{x}_i) \right\rangle. \tag{9–6.4}$$

The brackets indicate an averaging procedure that smoothes out the discreteness, generally some form of area- or volume-weighting over finite-sized particles among several cells. These averages and the algorithms that define the forces distinguish the various methods and approaches.

When the number of quasiparticles becomes very large, and a long-range interaction law is used, it is no longer practical to treat the forces by calculating direct interactions. A potential ϕ can be introduced,

$$\mathbf{f}_j = -\nabla\phi(\mathbf{x}_j). \tag{9–6.5}$$

This potential satisfies a Poisson equation relating the source terms, such as mass, to the corresponding long-range potential fields ϕ. This field ϕ is found by methods discussed in Chapter 10. Equations (9–6.1) and (9–6.2) are ordinary differential equations, for which many solution methods were described in Chapter 5.

When each quasiparticle has a fluid velocity that is determined by its position at any time,

$$\mathbf{v}_j = \mathbf{v}\big(\mathbf{x}_j(t),\ t\big), \tag{9–6.6}$$

such as in the case of vorticity transport considered in Section 9–7, the quasiparticles cannot pass through each other. When they do pass each other, it is the unphysical result of finite timestep errors or singularities in the interaction laws forming $\mathbf{v}(\mathbf{x}, t)$. The model is only appropriate for equilibrium, collisional fluids.

Incompressibility really arises only "on average" in real fluids. Thus the incompressibility condition, $\nabla \cdot \mathbf{v} = 0$, is difficult to enforce rigorously in quasiparticle models. Because

there are so many fewer quasiparticles in the model than real particles in the fluid, the statistical fluctuations about constant density can be quite large. One way to enforce incompressibility is to assume the density is constant, as in the vortex-dynamics and vortex-in-cell approaches discussed in Section 9–7.

Among the first quasiparticle methods developed for fluid dynamics are the *particle-in-cell* method (PIC) (Evans and Harlow 1957; Harlow 1964), the *marker-and-cell* method (MAC) (Harlow and Welch 1965), and the *grid-and-particle* method (GAP) (Marder 1975). These methods all use a finite-difference grid to obtain certain derived continuum quantities, such as the pressure, while using averaged particle values of the mass, momentum, and energy to preserve the Lagrangian features of the flow.

The original PIC method used a rectilinear Eulerian mesh. A mean velocity, density, internal energy, and pressure are defined within each computational cell. In the first stage of the calculation, the macroparticles are regarded as fixed. The density $\rho_{\alpha i}$ at a computational cell i, for a species α, is the mass m_α times the number of macroparticles of species α within the given cell, divided by the volume associated with the cell. The pressure is then calculated from the partial pressures $P_{\alpha i}$ derived from $\rho_{\alpha i}$ using a suitable equation of state. Then equations (9–1.14) and (9–1.15), or an analogous equation for the internal energy density ϵ, are solved without the advection terms to obtain intermediate values of $\mathbf{v}$ and E (or ϵ) on the grid. The equations are differenced to conserve momentum and energy. The energy is obtained by using the average of the old and the intermediate velocities in the compression term. The quasiparticles are then moved in a Lagrangian manner using the area-weighted intermediate velocities. Since the quasiparticles always have distinct locations, the numerical diffusion that would be associated with a donor-cell interpolation of Eulerian convection across the grid is partly avoided. Some of the quasiparticles cross cell boundaries and thus transfer mass, momentum, and energy convectively from one cell to another. This results in updated velocities, energy densities, and mass densities on the grid.

As with all quasiparticle methods, PIC has a graininess problem when the number of particles of given species per computational cell is small. In addition, the Eulerian differencing scheme in PIC is unstable. The method works because there is an effective viscosity, $\overline{\mu} \approx \frac{1}{2}\rho\overline{v}\Delta x$, in the calculation associated with particles crossing grid boundaries. This gives some effectively first-order donor-cell smoothing. Near stagnation regions, where the velocity approaches zero, the results are inaccurate. There are also problems associated with extreme expansions and weak shocks, where an additional artificial viscosity is needed.

MAC (Harlow and Welch 1965) is based on an Eulerian finite-difference method for incompressible flow in which marker particles are advected passively by the computed flow field. The markers participate in the dynamics by determining the position of a surface or interface, when that is required by the physics of the problem being solved. MAC was used extensively in surface water-wave problems, and is mentioned again in Chapter 10 in the discussion of interface tracking. MAC's main importance to the evolution of CFD is the staggered grid used to solve the pressure and velocity equations for incompressible flow. This innovation removed the spurious numerical instabilities that had plagued previous attempts to solve incompressible flow, and thus it was a forerunner of many modern structured and unstructured grid algorithms.

GAP (Marder 1975) was developed specifically to remove some of the problems with PIC using grid-oriented techniques developed primarily for plasma simulation. GAP differs from PIC in that the macroparticles carry the fluid properties with them, including specific volume, so numerical diffusion is significantly reduced. Area weighting is used to determine P and ρ on a fixed grid. These grid pressure values are then used to calculate the forces on the macroparticles. This avoids the near-neighbors problem of finding forces due to neighboring macroparticles and the grid provides a partial smoothing of the inherent graininess of the representation. A drawback, at least from the purist's point of view, is the presence of the fixed grid and the lack of Galilean invariance this implies. GAP treats shocks of any strength without requiring artificial viscosity and has better resolution and stability properties than PIC.

Lagrangian quasiparticle-dynamics methods, called direct-simulation Monte Carlo (DSMC), have been developed for the flow regime called either the transitional-flow or rarefied-gas regime (Bird 1994). As described in Chapter 1, the useful dimensionless number for characterizing this regime is the Knudsen number (Kn), defined as the ratio of the particle mean free path to a characteristic length scale of the system, $Kn = \lambda/L$. In the rarefied-flow regime, where the Knudsen number lies between 0.1 and 10, the Navier-Stokes equations are a poor model of the flow physics. Noncontinuum effects can be detected in flows down to $Kn = 0.01$. DSMC requires a grid, but it differs from those discussed previously in that the macroparticle interactions are modeled by random collisions between constituent particles. There is a simulated-to-actual particle ratio used to define the fluid properties. DSMC methods can be used for a range of rarefied gas problems, including complex reactive flows around spacecraft, rockets and rocket plumes, and planetary atmospheres. Most recently, DSMC methods have been applied to gas flows in micron-sized devices (Oran, Oh, and Cybyk 1998). They have also been very successfully extended to reactive flows with complex chemical reactions (see, for example, Bird (1994) and the extensive summaries of the Symposia on Rarefied Gas Dynamics, such as Harvey and Lord [1994]). A number of the adaptive gridding methods described in Chapter 6 have been tested for DSMC.

Quasiparticle methods are generally stable and conservative. They can also guarantee positivity and can be made reasonably general and flexible. Their drawback, as noted repeatedly, is a relative lack of efficiency and accuracy. Because the graininess and accuracy of the computed solutions depend on the number of macroparticles that the user can afford, many applications of interest are inaccessible to quasiparticle approaches. For example, three-dimensional problems are generally very expensive and the linear growth of fluid instabilities is difficult to compute and diagnose.

9–6.2. Smooth-Particle Hydrodynamics

Smooth-particle hydrodynamics techniques, sometimes called *smoothed*-particle hydrodynamics (Monaghan 1992), are a logical extension of decades of work on particle methods discussed above. SPH was started for astrophysical problems (Lucy 1977; Gingold and Monaghan 1977) and was soon adapted to other applications, primarily for computing the dynamic, fluid-like response of materials. Such a "gridless" Lagrangian method has potential advantages over gridded finite-volume and finite-element methods for computations

of dynamic fracture, impact, and fragmentation. An overview of recent developments is given by Randles and Libersky (1996).

SPH, as distinct from other particle approaches, uses interpolation techniques to smooth the statistical fluctuations inherent in particle methods. Because SPH is inherently grid free, there are no mesh or line crossings to untangle. The approach is almost as simple in three dimensions as in one dimension. It is intrinsically conservative because conserved quantities such as mass, momentum, and energy are defined with the smoothed macroparticles. It is also claimed that SPH algorithms are much more efficient for a given accuracy than standard gridded CFD representations (Stellingwerf 1991), a claim that is questionable. For example, when SPH was applied to standard one-dimensional shock test problems (Cloutman 1991 [Figure 2], Monaghan 1997a [Figures 1 to 7]), considerably more particles are required to achieve solutions to a given resolution and accuracy than cells used by the best continuum methods (described previously in this chapter and Chapter 8).

A main problem in macroparticle models is the need to derive relatively smooth, continuum fluid properties from the discrete particles and the associated quantities they carry. SPH does this by defining an interpolating kernel $W_{ij}(\mathbf{r}_i - \mathbf{r}_j, h)$ to smooth the determination of fluid-dynamic quantities. Here the subscripts i and j label two particles and h is a scale factor defining the extent of the generally nonnegative kernel. The kernal W_{ij} is localized in space, as defined by h, for example a truncated Gaussian, and normalized to unit integral. The value of a fluid-dynamic quantity, such as the density $\rho(\mathbf{r})$, is found by convolving the kernel function with the discrete SPH macroparticles, each carrying a mass, vector velocity, and other properties such as internal energy. The result is a sum smoothed by the kernel,

$$\rho(\mathbf{r}) = \sum_i m_i \, W(\mathbf{r} - \mathbf{r}_i, h). \tag{9–6.7}$$

Broader kernels result in smoother fluid-dynamic quantities, but with correspondingly more particle-particle interactions that must be treated. When $\mathbf{r} = \mathbf{r}_j$, equation (9–6.7) gives the density at the location of particle j. The kernel $W(\mathbf{r} - \mathbf{r}_i, h)$ can be viewed as a density distribution for the macroparticle at $\mathbf{r}_i$, and equation (9–6.7) defines a density at each point using the instantaneous locations of the particles.

It is also possible to advance the density at the particle locations by a hybrid continuity equation,

$$\frac{d\rho(\mathbf{r}_j)}{dt} = \sum_i m_i(\mathbf{v}_j - \mathbf{v}_i) \cdot \nabla_j W_{ji}, \tag{9–6.8}$$

(Monaghan 1997a). Similarly, the velocity of macroparticle j can be updated from

$$\frac{d\mathbf{v}(\mathbf{r}_j)}{dt} = -\sum_i m_i \left(\frac{P_j}{{\rho_j}^2} + \frac{P_i}{{\rho_i}^2} + \Pi_{ji} \right) \nabla_j W_{ji}, \tag{9–6.9}$$

where P_i and P_j are pressures at particle i and j, respectively. The term Π_{ji} describes the macroparticle equivalent of viscosity and can be extended to include artificial viscosity to improve the treatment of shocks (Monaghan 1992, 1997a; Libersky and Petschek 1991).

Derivatives of fluid quantities, such as the pressure gradients in equation (9–6.9), can be evaluated analytically by applying the gradient operator to the known, smooth kernel functions.

How the energy equation is treated depends on the particular application. This is one of the strengths of the overall SPH approach. Primary applications for SPH are to problems where the strength of material is important, and large-scale deformation leads to fracture and material separation. SPH is really a framework for combining particle-like and fluid properties for complex physics problems such as multiphase flows (see Monaghan [1997b]).

The disadvantages of the SPH methodology are apparent from the discussion of macroparticle representations given above. Because pure SPH models are grid free, there is the usual near-neighbors scaling problem that has to deal with computing interactions between all particle pairs. It has been found difficult to implement general material boundary conditions for complex geometry, but this is probably not an inherent failing of the methodology. The density, for example, is determined by summing over particles, and determining boundary values requires ghost particles or other analytic artifices. The macroparticles also interpenetrate when they should not, a situation largely handled by artificial-viscosity terms. The statistics of the finite number of macroparticles leads to a limited range of density values that can be handled accurately without problem-specific fixes. The size h of the kernels, for example, need not be the same for all particles and can even change in time. These generalizations add appreciable additional complexity.

SPH methods have also suffered a problem called "tension instability" (Swegle et al. 1994), a problem that appears to be related to that of spurious modes in continuum fluid models. Approaches for dealing with this have been proposed by Wen, Hicks, and Swegle (1994) who add dissipation and by Dyka and Ingel (1995) who change the weighting factors in the algorithm.

9–6.3. Lattice-Gas (Cellular) Automata

The quasiparticle methods described above compute continuum macroparticle positions and velocities. They attempt to improve the intrinsic graininess by using fluid-like interaction laws or interpolations that are designed to be insensitive to the local macroparticle number density. More complex quasiparticle models have generally given better results at the price of more complicated programming and slower simulations. An alternative approach involves abstracting the underlying particle nature of fluid dynamics even further by considering simpler, rather than more complex, particle models. The terms *lattice gas* or *lattice-gas automata* (*LGA*), have been coined for this class of models that fall under the general classification of *discrete-state cellular automata*. These relatively new approaches are efficient computationally, but they rely on very many degrees of freedom to reduce the numerical graininess to acceptable levels. The simulation particles do not interact through a continuum of collision cross-sections or have a continuum of velocities or positions.

The basic idea of LGA is to represent the fluid as a distribution of low-order interacting bits, or lattice-gas "atoms," that can reside only on the sites of a regular lattice. Both rectangular (d'Humières, Lallemand, and Frisch 1986) and triangular lattices (Frisch,

Hasslacher, and Pomeau 1986) have been used in both two and three dimensions. In the case of a three-dimensional rectangular lattice, an atom can have one of six different directions at each site, or it can be at rest. There is a rule that at most seven atoms can reside at each site, and there is an exclusion principal that says that, at most, one atom with a given direction can reside at the site. Atoms that are not at rest move one lattice link in their indicated direction at each timestep. Collisions are treated after all the particles are moved by randomly replacing the seven-bit state vector at each site with a another physically allowed state vector. Mass and energy are conserved in these collisions by ensuring, respectively, that the number of atoms and the number of moving atoms at each site do not change. Momentum is conserved by ensuring that the number of atoms moving in each component direction does not change.

One of the attractive features of the model is its inherent simplicity. Efficient implementation of the algorithms is very machine dependent, and special computer chips have been constructed implementing specific algorithms. Complex geometry can be included simply by blocking out some of the links. In the seven-state automata for two dimensions described above, this can be accomplished by simply setting an eighth bit and using a full byte to describe the lattice. The advantage of adopting a model with more directions for the lattice links (e.g., one that is triangular rather than rectangular) is an improved isotropy of the fluid dynamics that results from the statistical mechanics of the model behavior.

Considerable development of LGA models has led to rather complex models and a broadening range of capabilities. For example, a lattice Boltzmann approach (Kadanoff, McNamara, and Zanetti 1989; Succi, Foti, and Higuera 1989; Chen and Doolen 1998) has been developed to cope with the inherent graininess of particle models by solving for the (continuous) velocity distributions at each lattice site. Partial differential equations are now solved, but some of the simplicity of the original method is lost. As with all particle and quasiparticle models, the veracity improves with the model complexity, implying a complicated trade-off that the developer and user have to consider. Adding more atom velocities (e.g., zero, one, or two) can improve the flexibility at the cost of considerable additional program complexity.

There are other limitations that also make LGA models difficult to recommend for practical reactive-flow applications. They are statistically grainy, a problem they share with all particle models for fluid dynamics. Since each moving atom is traveling at about the sound speed, LGA represents fluid dynamics correctly only in the limit of zero Mach number. This means that the error increases as the average momentum in the vicinity of each site increases. LGA models generally cannot represent a wide range of densities accurately since a finite number of particles is used and they occupy a rigid lattice. If the density is very low in a region, the graininess gets worse. For reactive flows there is a problem shared with other particle and quasiparticle methods: there is no good way to represent trace species, such as reactive radicals, that may control the evolution of the system and whose density is orders of magnitude below the average density. Finally, the computational work for LGA models does not scale as well with increasing Reynolds number, as is the case for Navier-Stokes equation solvers (Orszag and Yakhot 1986). Karniadakis et al. (1993) review LGA and compare the computed results to those obtained by spectral element models.

9–7. Vortex and Contour Dynamics

Vortex dynamics and contour dynamics are based on an alternate Lagrangian representation to the usual pressure-velocity formulation given by equations (9–1.4) through (9–1.7). In these representations, the vorticity field is represented by a set of discrete macroparticles that describe filaments of circulation. These models take advantage of the facts that vorticity is convected with the flow and the circulation of a vortex filament is generally conserved. The velocity field is calculated from the complete set of macroparticles. The resulting equations that have to be solved are a set of nonlinear ordinary differential equations describing the time evolution of the discrete vortex locations. These representations have the advantage of needing no degrees of freedom where there is no vorticity. They are, however, best used as algorithms for incompressible flows in geometrically simple domains.

9–7.1. Vortex Dynamics

The earliest vortex-dynamics method (Rosenhead 1931) considered the time-dependent behavior of a two-dimensional vortex sheet represented as a system of point vortices. In two dimensions the scalar vorticity field can be written as a sum

$$\omega(\mathbf{r}, t) = \sum_{i=1}^{N} \Gamma_i \delta\big(\mathbf{r} - \mathbf{r}_i(t)\big), \tag{9–7.1}$$

where δ is the two-dimensional Dirac delta function, and the vectors $\mathbf{r}_i = (x_i, y_i)$ are the locations of N singular point vortices. The $\{\Gamma_i\}$ are the circulations (or strengths) of each vortex, defined as

$$\Gamma_S = \oint_S \omega \, dS. \tag{9–7.2}$$

The constant density, vorticity-transport equation, equation (9–1.10) with an added term to describe the diffusion of vorticity caused by viscosity, is

$$\frac{d\omega}{dt} = \frac{\partial \omega}{\partial t} + (\mathbf{v} \cdot \nabla)\omega = \nu \nabla^2 \omega. \tag{9–7.3}$$

For the inviscid case, $\nu = 0$ in equation (9–7.3), these discrete vortices must be convected conservatively with the flow. The velocity of each vortex is taken as the instantaneous velocity due to all the other point vortices evaluated at the vortex location,

$$\frac{d\mathbf{r}_i}{dt} = \mathbf{v}(\mathbf{r}_i, t). \tag{9–7.4}$$

If the two-dimensional flow field has no interior boundaries and the fluid is at rest at infinity, the driving velocity in equation (9–7.4) may be written as a Biot-Savart integral. Then using equation (9–7.1) for the vorticity source term, the evolving vortex positions $\{\mathbf{r}_i\}$ are solutions to the following set of nonlinear ordinary differential equations,

$$\frac{d\mathbf{r}_i}{dt} = -\frac{1}{2\pi} \sum_{j \neq i}^{N} \frac{(\mathbf{r}_i - \mathbf{r}_j) \times \hat{z}\, \Gamma_j}{|\mathbf{r}_i - \mathbf{r}_j|^2}. \tag{9–7.5}$$

When $\nabla \cdot \mathbf{v} = 0$, the instantaneous distribution of vorticity is the principal source of flow. In this case, the velocity field can be reconstructed from the vorticity through the solution of the general div-curl problem. Here equation (9–7.5) gives a prescription for determining the velocity field without actually solving the elliptic equation for the stream function ψ given in equation (9–3.5). In three dimensions, these equations can be generalized to describe the mutual interaction and convection of filaments of vorticity.

The point-vortex method is not stable and does not generally converge, in part because equation (9–7.5) becomes singular when two point vortices approach each other. An increasing number of point vortices of reduced strength with more accurate integration methods does not produce a converged solution. The problem is that treating points instead of distributed vorticity produces singularities in the solution after a finite time. *Distributed-core* vortex methods attempt to remedy this problem by smearing the vorticity over a finite radius core (Chorin 1973; Chorin and Bernard 1973). Spreading the vortex cores gives a smoother representation of the vorticity, and the induced velocities in and near vortex elements are now bounded. Various types of core distribution functions have been tried. The Gaussian core distribution, for example, is second-order accurate. Higher-order accuracy can be obtained by specially constructing the core distribution (Hald and Del Prete 1978; Hald 1979). Quite complicated flow physics such as spatially evolving shear layers (Ghoniem and Ng 1987) have been integrated using vortex dynamics. There has also been work to extend the method to somewhat compressible flows.

There are a number of ways of calculating the velocity for a set of distributed-core vortices. In all cases, advancing the model requires solving a coupled set of nonlinear ordinary differential equations. Remaining problems with accuracy in the distributed-core vortex method are associated with the assumption of a constant shape function for the vortex elements at all times. A real fluid element with a finite amount of distributed circulation would become strained and distorted. One way to test the accuracy is to evaluate the integral constraints on the motion which should not change in time, (see [Leonard 1980]).

It is important to be able to add viscosity to these methods. Viscosity appears in the vorticity diffusion term, $\nu \nabla^2 \omega$, in the vorticity transport equation (9–7.3). Diffusion of vorticity can be treated by allowing the Gaussian vortex cores of radius σ to increase in size according to

$$\frac{d\sigma^2}{dt} = 4\nu. \tag{9–7.6}$$

Another approach proposed by Chorin (1973) is to add a random walk each timestep, with a spatial displacement proportional to $(\nu \Delta t)^{\frac{1}{2}}$ (see also [Ghoniem and Sherman 1985]). These *random-vortex methods* have the advantage that the vortices can all stay the same size. They have the potential disadvantage, compared to fully deterministic vortex methods implementing equation (9–7.6), that additional statistical fluctuations are steadily being added to a flow that should become smoother with time because of viscosity, not rougher.

Distributed-filament vortices are the generalization of distributed-core vortices to three dimensions. Here it is assumed that the vorticity field is represented by a collection of N filaments. Because the vorticity is divergence free, the circulation along a filament can be considered constant. A major complication, however, is that the radial extent of filament i, σ_i, varies with distance along the filament. Because the filament maintains its integrity as it

is moved, an average velocity must be chosen to move the filament. Again, there are several ways to do this in analogy with the two-dimensional distributed-core vortex methods. For viscous effects, the profile associated with σ_i and Γ_i can be modified. Leonard (1980) points out that for flows with a low Reynolds number (large ν), this should work well. For flows with a high Reynolds number, the curve $\mathbf{r}_i$ can be distorted greatly in space due to large velocity gradients. These turbulent effects can be modeled by introducing an eddy-viscosity approximation or a more sophisticated model (see, for example, Knio and Ghoniem [1991, 1992]).

9–7.2. Vortex-in-Cell Methods

Direct evaluation of the velocity integrals is required in the point-vortex method, as indicated in equation (9–7.5). The distributed-core vortex and filament methods require $\mathcal{O}(N^2)$ operations when there are N elements. Vortex-in-cell methods still integrate the vorticity in a Lagrangian representation, but they solve a Poisson equation at every timestep for the velocity on a grid of M points to avoid the N^2 problem. This Poisson equation defines the vector stream function $\boldsymbol{\psi}$ of equation (9–1.5), (9–1.6), and (9–3.5), given the vorticity $\boldsymbol{\omega}$ as a source term. This solution requires only $\mathcal{O}(M \ln M)$ operations. Grid values for the vorticity must be built up from the Lagrangian locations of all the vortices, and the corresponding velocities must be interpolated from the grid back to the locations of the Lagrangian vortices. The resulting algorithms typically have a total operation count of $\mathcal{O}(M + M \ln M)$. Thus, there is a large gain if M is kept small enough relative to N. This approach was proposed by Roberts and Christiansen (1972) and first implemented by Christiansen (1973) in two dimensions using the bilinear "cloud-in-cell" interpolation reviewed in Birdsall and Langdon (1985). Subsequently the vortex-in-cell method was applied by Christiansen and Zabusky (1973) to uniform vorticity regions and demonstrated in three dimensions by Couët, Buneman, and Leonard (1981).

The problem with this method is that a grid is introduced, partly obviating the advantages of a Lagrangian model. This grid does, however, allow the model to be solved quickly, even on parallel-processing systems and avoids the singularities of the point-vortex description. As with the class of incompressible flow methods described in Section 9–3, it is difficult and expensive to solve the Poisson equation in complex geometry at every timestep. It is even more difficult and expensive to evaluate the Biot-Savart integrals every timestep to obtain the velocity from the vorticity distribution.

9–7.3. Contour Dynamics

Contour dynamics is principally a two-dimensional method in which regions of piecewise-constant vorticity are enclosed by contours (Deem and Zabusky 1978; Zabusky, Hughes, and Roberts 1979; Zabusky and Overman 1983; Buttke 1990). If the vorticity in a region is initially constant, it stays that way under inviscid motion, but each region is distorted as it is strained by the velocity field. Contour dynamics calculates the evolving boundaries of regions of constant vorticity. In principle, a continuous vorticity distribution can be approximated by a number of contours. Since an evolution equation for the curves that separate the different constant-vorticity regions can be written as a line integral along the curves, contour dynamics has a lot in common with boundary-integral methods.

To track these contours, points or nodes are defined on the curves, and the motions of these points are integrated moving with the flow. When the boundary curves between regions of constant vorticity form singularities, cusps, or filaments, and when the total length of the curves increases in time, it is necessary to redistribute the points along the curves and often to add additional points to maintain the accuracy of the representation. These remedial procedures have been called "contour surgery." Small-scale features that have little effect on the overall flow evolution can be removed or averaged out, a process somewhat analogous to a structure passing into a coarser grid in a finite-difference calculation. The expectation is that this process does not influence the large-scale structures appreciably.

A number of different representations and algorithms are available for specifying and integrating the contour dynamics. Legras and Zeitlin (1992) propose a method based on mapping from a standard domain onto the domain of the vortex dynamics contours and give a dynamic equation for their evolution. VanBuskirk and Marcus (1994) present an exponentially accurate boundary-integral method for evolving the dynamics of piecewise-constant distributions vorticity. As with vortex dynamics, if the velocity is computed directly as an integral over the contours, contour dynamics scales as N^2, where N is the number of nodes defining the contours. Dritschel (1993) describes an approach called "moment-accelerated contour surgery" that uses an alternative expression for the velocity field, one that takes the form of a rapidly convergent series, in the exterior of a vortex. Schmitt (1994) then showed that the fast multipole method can be applied directly to give an algorithm that scales as N, with correspondingly large decrease in required computational time. Similar techniques might be used to speed up vortex dynamics.

There are two lessons to be learned from contour dynamics. First, a continuous distribution of vorticity more nearly describes many physical situations than a collection of discrete vortices. Second, rotational vortex flow necessarily involves deformation, even at the small scale. By understanding these issues and building the representation to deal with them, contour dynamics has produced some of the most accurate computations that have been performed of one class of fluid-dynamics problems (see the review by Pullin [1992]). Nonetheless, as in the case for vortex dynamics, complex physics (such as complex geometry, three-dimensional flow, and reactive flow) are difficult to include properly.

9–7.4. Some Comments on Vortex Methods

Excellent summaries of vortex methods have been given by Leonard (1980, 1985) and Puckett (1993) and they form a good basis for going beyond the brief discussion presented here. Incompressible flows with highly localized vorticity show vortex dynamics and contour dynamics to their best advantage. In these methods, however, the number of operations per timestep is $\mathcal{O}(N^2)$, and this can be expensive unless a grid and some form of elliptic solver is reintroduced. Vortex-dynamics and contour-dynamics methods can automatically focus on developing small-scale structure by locally concentrating points, although carrying many Lagrangian nodes is expensive. There are methods for treating boundary conditions at infinity and for outflow, but no-slip boundary conditions to account for vorticity generation at a solid wall are difficult to implement elegantly. Leonard also describes and references a number of applications of these methods, such as simulations of incompressible turbulence and certain boundary-layer flows.

A point of current popular discussion is the application of these methods to reactive flows, particularly to combustion where it is necessary to resolve expansions, compressions, and even the effects of sound waves. This cannot be done in a pure vortex-dynamics or contour-dynamics method, which is basically incompressible. For a complete solution, it is necessary to introduce a grid and solve a Poisson equation for the pressure, as is done with the vortex-in-cell methods. Vortex dynamics has been combined with an interface-tracking method for the flame front and used to simulate the behavior of flames (see, for example, [Ashurst and Barr 1983; Barr and Ashurst 1984]). Methods for adding more complex physics, as is necessary for reactive flows, are reviewed by Ghoniem (1991). Because of the complexity and the near impossibility of an efficient implementation for complex geometry, however, these methods have not gained a large following for reactive flow.

REFERENCES

Ashurst, W.T., and P.K. Barr. 1983. Stochastic calculation of laminar wrinkled flame propagation via vortex dynamics. *Combustion Science and Technology* 34:227–256.

Barr, P.K., and W.T. Ashurst. 1984. *An interface scheme for turbulent flame propagation*. Sandia report SAND82-8773. Albuquerque, N. Mex.: Sandia National Laboratories.

Baum, J.D., H. Luo, and R. Löhner. 1994. *A new ALE adaptive unstructured methodology for the simulation of moving bodies*. American Institute for Aeronautics and Astronautics paper AIAA-94-0414. Reston, Va.: AIAA.

Beam, R.M., and R.F. Warming. 1976. An implicit finite-difference algorithm for hyperbolic systems in conservation-law form. *Journal of Computational Physics* 22:87–110.

———. 1978. An implicit factored scheme for the compressible Navier-Stokes equations. *AIAA Journal* 16:393–401.

Bell, J.B., P. Colella, and H.M. Glaz. 1989. A second-order projection method for the incompressible Navier-Stokes equations. *Journal of Computational Physics* 85:257–283.

Bell, J.B., and D.L. Marcus. 1992. A second-order projection methods for the incompressible Navier-Stokes equations. *Journal of Computational Physics* 101:334–348.

Bertagnolio, F., and O. Daube. 1997. Solution of the div-curl problem in generalized curvilinear coordinates. *Journal of Computational Physics* 138:121–152.

Bird, G.A. 1994. *Molecular gas dynamics and the direct simulation of gas flows*. Oxford, England: Oxford Univ. Press.

Birdsall, C.K., and D. Fuss. 1969. Clouds-in-clouds, clouds-in-cells physics for many-body plasma simulation. *Journal of Computational Physics* 3:494–511, reprinted in the 30th anniversary edition: *Journal of Computational Physics* 135:141–148, 1997.

Birdsall, C.K., and A.B. Langdon. 1985. *Plasma physics via computer simulation*. New York: McGraw-Hill.

Börgers, C., and C.S. Peskin. 1985. A Lagrangian method based on the Voronoi diagram for the incompressible Navier-Stokes equations on a periodic domain. In *The free-Lagrange method, lecture notes in physics* 238, eds. M.J., Fritts, W.P. Crowley, and H. Trease, 87–113. New York: Springer-Verlag.

Boris, J.P. 1970a. *Relativistic plasma simulation – Optimization of a hybrid code*. In *Proceedings, Fourth Conference on the Numerical Simulation of Plasma*, 3–67. Washington, D.C.: U.S. Government Printing Office.

———. 1970b. *A physically motivated solution of the Alfven problem*. Memorandum report 2167. Washington, D.C.: Naval Research Laboratory.

———. 1979. *ADINC: An implicit Lagrangian hydrodynamics code*. NRL memorandum report 4022. Washington, D.C.: Naval Research Laboratory.

———. 1986. A vectorized "near neighbors" algorithm of order N using a monotonic Lagrangian grid. *Journal of Computational Physics* 66:1–20.

Briley, W.R., and H. McDonald. 1975. Solution of the three-dimensional compressible Navier Stokes equations by an implicit technique. In *Proceedings Fourth International Conference on Numerical Methods in Fluid Dynamics*, ed. R.D. Richtmyer, 105–110. New York: Springer-Verlag.

———. 1977. Solution of the multi-dimensional compressible Navier-Stokes equations by a generalized implicit method. *Journal of Computational Physics* 24:372–397.

———. 1980. On the structure and use of linearized block implicit schemes. *Journal of Computational Physics* 34:54–73.

Bruel, P., D. Karmed, and M. Champion. 1996. A pseudo-compressibility method for reactive flows at zero Mach number. *International Journal of Computational Fluid Dynamics* 7:291–310.

Buneman, O. 1967. Time-reversible difference procedures. *Journal of Computational Physics* 1:517–535.

Buttke, T.F. 1990. A fast adaptive vortex method for patches of constant vorticity in two dimensions. *Journal of Computational Physics* 89:161–186.

Caramana, E.J., and M.J. Shashkov. 1998. Elimination of artificial grid distortion and hourglass-type motions by means of Lagrangian subzonal masses and pressures. *Journal of Computational Physics* 142:521–561.

Casulli, V., and D. Greenspan. 1984. Pressure method for the numerical solution of transient, compressible fluid flows. *International Journal of Numerical Methods for Fluids* 4:1001–1012.

Cerutti, J.H., and H.E. Trease. 1991. The free-Lagrange method on the connection machine. In *Advances in the free-Lagrange method, lecture notes in physics* 395, eds. H. Trease, M.J. Fritts, and W.P. Crowley 183–192. New York: Springer-Verlag.

Chen, S., and G.D. Doolen. 1998. Lattice Boltzmann method for fluid flows. *Annual Reviews of Fluid Mechanics* 30:329–364.

Chorin, A.J. 1967. A numerical method for solving incompressible viscous flow problems. *Journal of Computational Physics* 2:12–26, reprinted in the *Journal of Computational Physics* 135:118–125, 1997.

———. 1968. Numerical solution of the Navier-Stokes equations. *Math. Comp.* 22:745–762.

———. 1973. Numerical study of slightly viscous flow. *Journal of Fluid Mechanics* 57:785–796.

———. 1976. Random choice solution of hyperbolic systems. *Journal of Computational Physics* 22:517–534.

———. 1977. Random choice methods with applications to reacting gas flows. *Journal of Computational Physics* 25:253–273.

Chorin, A.J., and P.S. Bernard. 1973. Discretization of a vortex sheet, with an example of roll-up. *Journal of Computational Physics* 13:423–429.

Christiansen, J.P. 1973. Numerical simulation of hydrodynamics by the method of point vortices. *Journal of Computational Physics* 13:363–379, reprinted in the 30th anniversary edition, *Journal of Computational Physics* 135, 189–197, 1997.

Christiansen, J.P., and N.J. Zabusky. 1973. Instability, coalescence and fission of finite-area vortex structures. *Journal of Fluid Mechanics* 61:219–243.

Cloutman, L.D. 1991. An evaluation of smoothed particle hydrodynamics. In *Advances in the free-lagrange method, Lecture Notes in Physics* 395, eds. H.E. Trease, M.J. Fritts, and W.P. Crowley. New York: Springer-Verlag.

Colella, P., and P.R. Woodward. 1984. The piecewise parabolic method (PPM) for gas-dynamical simulations. *Journal of Computational Physics* 54:174–201.

Cooper, A., J.M. Pierre, P.J. Turchi, J.P. Boris, and R.L. Burton. 1980. Modeling of LINUS-type stabilized liner implosions. In *Megagauss physics and technology*, ed. P.J. Turchi, 447–460. New York: Plenum.

Couët, B., O. Buneman, and A. Leonard. 1981. Simulation of three-dimensional incompressible flows with a vortex-in-cell method. *Journal of Computational Physics* 39:305–328.

Courant, R., and K.O. Friedrichs. 1948. *Supersonic flow and shock waves*, New York: Interscience.

Courant, R., and D. Hilbert. 1962. *Methods of mathematical physics*. Vol. II. New York: Interscience.

Courant, R., E. Isaacson, and M. Reeves. 1952. On the solution of nonlinear hyperbolic differential equations by finite differences. *Communications in Pure and Applied Mathematics* 5:243–255.

Crowley, W.P. 1985. Free-Lagrange methods for compressible hydrodynamics in two space dimensions.

In *The free-Lagrange method, Lecture notes in physics* 238, eds. M.J. Fritts, W.P. Crowley, and H. Trease, 1–21. New York: Springer-Verlag.

d'Humières, D., P. Lallemand, and U. Frisch. 1986. Lattice gas model for 3D hydrodynamics. *Europhysics Letters* 2:291–297.

Deem, G.S., and N.J. Zabusky. 1978. Vortex waves: Stationary "V" states, interactions, recurrence and breaking. *Physical Review Letters* 40:859–862.

Donea, J., S. Guilani, H. Laval, and L. Quartepelle. 1982. Solution of the unsteady Navier-Stokes equations by a fractional step method. *Comp. Math. Applied Mechanical Engineering* 30:53–73.

Dritschel, D.G. 1993. A fast contour dynamics method for many-vortex calculations in two-dimensional flows. *Physics of Fluids A* 5:173–186.

Dyka, C.T., and R.P. Ingel. 1995. An approach for tension instability in smoothed particle hydrodynamics (SPH). *Computers and Structures* 57:573–580.

Eidelman, S. 1986. Local cell orientation method. *AIAA Journal* 24:530–531.

Eltgroth, P.G. 1985. Compressible Lagrangian hydrodynamics without Lagrangian cells. In *The free-Lagrange method, lecture notes in physics* 238, eds. M.J. Fritts, W.P. Crowley, and H. Trease, 114–121. New York: Springer-Verlag.

———. 1991. Asynchronous 3D free-Lagrange code. In *Advances in the free-Lagrange method, lecture notes in physics* 395, eds. H.E. Trease, M.J. Fritts, and W.P. Crowley, 76–81. New York: Springer-Verlag.

Evans, M.W., and F.H. Harlow. 1957. *The particle-in-cell method for hydrodynamic calculations*. LASL report no. LA-2139. Los Alamos, N. Mex.: Los Alamos National Laboratory.

Frisch, U., B. Hasslacher, and Y. Pomeau. 1986. Lattice gas automata for the Navier-Stokes equation. *Physical Review Letters* 56:1505–1508.

Fritts, M.J. 1985. Three-dimensional algorithms for grid restructuring in free-Lagrangian calculations. In *The free-Lagrange method, lecture notes in physics* 238, eds. M.J. Fritts, W.P. Crowley, and H. Trease, 122–144. New York: Springer-Verlag.

Fritts, M.J., and J.P. Boris. 1979. The Lagrangian solution of transient problems in hydrodynamics using a triangular mesh. *Journal of Computational Physics* 31:173–215.

Fritts, M.J., W.P. Crowley, and H. Trease, eds. 1985. *The free-Lagrange method, lecture notes in physics* 238. New York: Springer-Verlag.

Fryxell, B.A., P.R. Woodward, P. Colella, and K.-H. Winkler. 1986. An implicit-explicit hybrid method for Lagrangian hydrodynamics. *Journal of Computational Physics* 63:283–310.

Fyfe, D.E., E.S. Oran, and M.J. Fritts. 1988. Surface tension and viscosity with Lagrangian hydrodynamics on a triangular mesh. *Journal of Computational Physics* 76:349–384.

Ghoniem, A.F. 1991. Vortex simulation of reacting shear flow. In *Numerical approaches to combustion modeling, progess in aeronautics and astronautics* 135, eds. E.S. Oran and J.P. Boris, 305–348. Washington, D.C.: AIAA.

Ghoniem, A.F., and K.K. Ng. 1987. Numerical study of the dynamics of a forced shear layer. *Physics of Fluids* 30:706–721.

Ghoniem, A.F., and F.S. Sherman. 1985. Grid free simulation of diffusion using random walk methods. *Journal of Computational Physics 61*:1–37.

Gingold, R.A., and J.J. Monaghan. 1977. Smoothed particle hydrodynamics: Theory and application to non-spherical stars. *Monthly Notices of the Royal Astronomical Society* 181:375–389.

Gittings, M.L. 1991. TRIX, A free-Lagrangian hydrocode. In *Advances in the free-Lagrange method, lecture notes in physics* 395, eds. H.E. Trease, M.J. Fritts, and W.P. Crowley, 28–36. New York: Springer-Verlag.

Glimm, J. 1965. Solutions in the large for nonlinear hyperbolic systems of equations. *Communications in Pure and Applied Mathematics* 18:697–715.

Godunov, S.K. 1959. Finite difference methods for numerical computation of discontinuous solutions of the equations of fluid dynamics. *Mat. Sb.* 47:271–306.

Grinstein, F.F., E.S. Oran, and J.P. Boris. 1990. Reinitiation and feedback in global instabilities of subsonic spatially developing mixing layers. *Physical Review Letters* 64:870–873.

———. 1991. Pressure field, feedback, and global instabilities of subsonic spatially developing mixing layers. *Physics of Fluids A: Fluid Dynamics* 3:2401–2409.

Gunzburger, M.D., and R.A. Nicolaides, eds. 1993. *Incompressible computational fluid dynamics*. New York: Cambridge University Press.

Hald, O.H. 1979. Convergence of vortex methods for Euler's equations, II. *SIAM Journal of Numerical Analysis* 16:726–755.

Hald, O.H., and V.M. Del Prete. 1978. Convergence of vortex methods for Euler's equations. *Mathematics of Computation* 32:791–809.

Harlow, F.H. 1964. The particle-in-cell computing method for fluid dynamics. In *Methods in computational physics* 3, eds. B. Adler, S. Fernback, and M. Rotenberg, 319–343. New York: Academic Press.

Harlow, F.H., and J.F. Welch. 1965. Numerical calculation of time-dependent viscous incompressible flow of fluid with free surface. *Physics of Fluids* 8:2182–2189.

Harten, A. 1983. High resolution schemes for hyperbolic conservation laws. *Journal of Computational Physics* 49:357–393, reprinted in the *Journal of Computational Physics* 135:260–278, 1997.

Harten, A., P.D. Lax, and B. Van Leer. 1983. On upstream differencing and Godunov-type schemes for hyperbolic conservation laws. *SIAM Review* 25:35–61.

Harvey, J., and G. Lord, eds. 1994. *Rarefield gas dynamics 19*, Oxford: Oxford University Press.

Hirt, C.W. 1971. An arbitrary Lagrangian Eulerian method. In *Proceedings of the Second International Conference on Numerical Methods in Fluid Dynamics*, ed. M. Holt. Berkeley, Berlin: Springer-Verlag.

Hirt, C.W., A.A. Amsden, and J.L. Cook. 1974. An arbitrary Lagrangian Eulerian computing method for all flow speeds. *Journal of Computational Physics* 14:227–254, reprinted in the *Journal of Computational Physics* 135:203–216, 1997.

Hirsch, C. 1990. *Numerical computation of internal and external flows, Vol. 2: Computational methods for inviscid and viscous flows*. New York: Wiley.

Hockney, R.W. 1965. A fast direct solution of Poisson's equation using Fourier analysis. *J. Comm. Assoc. Comp. Mach.* 12:95–113.

———. 1966. *Further computer experimentation on anomalous diffusion*. SUIPR report no. 202. Stanford, Calif.: Institute for Plasma Research, Stanford University.

Hoskin, N.E. 1964. Solution by characteristics of the equations of one-dimensional unsteady flow. In *Methods in Computational Physics* 3, eds. B. Adler, S. Fernback, and M. Rotenberg, 265–293. New York: Academic Press.

Jones, W.W., and J.P. Boris. 1977. Flame and reactive jet studies using a self-consistent two-dimensional hydrocode. *Journal of Physical Chemistry* 81:2532–2534.

———. 1979. *FLAME – A slow-flow combustion model*. NRL memorandum report, no. 3970, Washington, D.C.: Naval Research Laboratory.

Kadanoff, L.P., G.R. McNamara, G. Zanetti. 1989. From automata to fluid-flow – Comparisons of simulation and theory. *Physical Review A* 40:4527–4541.

Kailasanath, K. 1991. Laminar flames in premixed gases. In *Numerical approaches to combustion modeling*, eds. E.S. Oran and J.P. Boris, *Progress in Astronautics and Aeronautics* 135, 225–256. Reston, VA.: AIAA.

Karniadakis, G.E., S.A. Orszag, E.M. Ronquist, and A.T. Patera. 1993. Spectral element and lattice gas methods for incompressible fluid dynamics. In *Incompressible computational fluid dynamics*, eds. M.D. Gunzburger and R.A. Nicolaides, 203–266. New York: Cambridge University Press.

Khokhlov, A.M. 1998. Fully threaded tree algorithms for adaptive mesh refinement fluid dynamic simulations. *Journal of Computational Physics* 143:519–543.

Kim, C.A., L. Martinelli, A. Jameson, and K. Xu. 1997a. An accurate LED-BGK Solver on unstructured adaptive meshes, AIAA paper 97-0328. Reston, VA: American Institute of Aeronautics, and Astronautics.

Kim, J., and P. Moin. 1985. Appplication of a fractional step method to incompressible Navier-Stokes equations. *Journal of Computational Physics* 59:308–323.

Kim, C.A., K. Xu, L. Martinelli, and A. Jameson. 1997b. Analysis and implementation of the gas-kinetic

BGK scheme for computational gas dynamics. *International Journal for Numerical Methods in Fluids* 25:21–49.

Knio, O.M., and A.F. Ghoniem. 1991. Three-dimensional vortex simulation of rollup and entrainment in a shear layer. *Journal of Computational Physics* 97:172–223.

———. 1992. Vortex simulation of a three-dimensional reacting shear layer with infinite-rate kinetics. *AIAA Journal* 30:105–116.

Kulikovsky, A.G., and G.A. Lyubimov. 1965. *Magnetohydrodynamics*. Reading, Mass.: Addison-Wesley.

Lambrakos, S., and J.P. Boris. 1987. Geometric properties of the monotonic Lagrangian grid algorithm for near neighbors calculations. *Journal of Computational Physics* 73:183–202.

Landau, L.D., and E.M. Lifshitz. 1959. *Fluid mechanics*. New York: Pergamon Press.

Langdon, A.B., and B.J. Lasinski. 1976. Electromagnetic and relativistic plasma simulations models. In *Methods in computational physics* 16, eds. B. Adler, S. Fernbach, and J. Killeen, 327–366. New York: Academic Press.

Legras B., and V. Zeitlin. 1992. Conformal dynamics for vortex motions. *Physics Letters A* 167: 265–271.

Leonard, A. 1980. Vortex methods for flow simulation. *Journal of Computational Physics* 37:289–335.

———. 1985. Computing three-dimensional incompressible flows with vortex elements. *Annual Reviews of Fluid Mechanics* 17:523–559.

Libersky, L.D., and A.G. Petschek. 1991. Smooth particle hydrodynamics with strength of materials. In *Advances in the free-Lagrange method, lecture notes in physics* 395, eds. H.E. Trease, M.J. Fritts, and W.P. Crowley, 248–257. New York: Springer-Verlag.

Liepmann, H.W., and A. Roshko. 1957. *Elements of gasdynamics*. New York: Wiley.

Lindemuth, I., and J. Killeen. 1973. Alternating direction implicit techniques for two dimensional magnetohydrodynamics calculations. *Journal of Computational Physics* 13:181–208.

Löhner, R. 1993. Design of incompressible flow solvers: Practical aspects. In *Incompressible Computational Fluid Dynamics*, 267–294. New York: Cambridge University Press.

Löhner, R., and C. Yang. 1996. Improved ALE mesh velocities for moving bodies. *Communications in Numerical Methods in Engineering* 12:599–608.

Lucy, L.B. 1977. A numerical approach to the testing of the fission hypothesis. *Astrophysical Journal* 82:1013–1024.

Mandell, D.A., and H.E. Trease. 1989. Parallel processing a three dimensional free-Lagrange code: A case history. *International Journal of Supercomputer Applications* 3:92–99.

Marder, B.M. 1975. GAP – A PIC-type fluid code. *Mathematics of Computation* 24:434–436.

McDonald, H., and W.R. Briley. 1975. Three-dimensional supersonic flow of a viscous or inviscid gas. *Journal of Computational Physics* 19:150–178.

Monaghan, J.J. 1992. Smoothed particle hydrodynamics. *Annual Reviews of Astronomy and Astrophysics* 30:543–574.

———. 1997a. SPH and Riemann solvers. *Journal of Computational Physics* 136:298–307.

———. 1997b. Implicit SPH drag and dusty gas dynamics. *Journal of Computational Physics* 138:801–820.

Moretti, G., and M. Abbett. 1966. A time-dependent computational method for blunt body flows. *AIAA Journal* 4:2136–2141.

Morgan, K., and J. Peraire. 1998. Unstructured grid finite-element methods for fluid mechanics. *Reports on Progress in Physics* 61:569–638.

Mulder, W.A., and B. Van Leer. 1985. Experiments with implicit upwind methods for the Euler equations. *Journal of Computational Physics* 59:232–246.

Nicolaides, R.A. 1993. The covolume approach to computing incompressible flows. In *Incompressible Computational Fluid Dynamics*, 295–334. New York: Cambridge University Press.

Obayashi, S., and K. Fujii. 1985. Computation of three-dimensional viscous transonic flows with the LU factored scheme. In *AIAA 7th Computational Fluid Dynamics Conference*, 192–202. New York: AIAA.

Oran, E.S., C.K. Oh, and B.Z. Cybyk. 1998. Direct simulation Monte Carlo – Recent advances and applications. *Annual Reviews of Fluid Mechanics* 30:403–441.

Orszag, S.A., and V. Yakhot. 1986. Reynolds number scaling of cellular automata hydrodynamics. *Physical Review Letters* 56:1691–1693.

Paolucci, S. 1982. *On the filtering of sound from the Navier-Stokes equations*. Sandia report SAND82-8257. Albuquerque, N. Mex.: Sandia National Laboratories.

Patankar, S.V. 1980. *Numerical heat transfer and fluid flow*. Washington, D.C.: Hemisphere.

———. 1981. A calculation procedure for two-dimensional elliptic situations. *Numerical Heat Transfer* 4:409–425.

Patankar, S.V., and D.B. Spalding. 1972. A calculation procedure for heat, mass and momentum transfer in three-dimensional parabolic flows. *International Journal of Heat and Mass Transfer* 15:1787–1806.

Patera, A.T. 1984. A spectral-element method for fluid dynamics: Laminar flow in a channel expansion. *Journal of Computational Physics* 54:468–488.

Patnaik, G. n.d. *A modified barely implicit correction for flux-corrected transport*. Naval Research Laboratory memorandum report.

Patnaik, G., R.H. Guirguis, J.P. Boris, and E.S. Oran. 1987. A barely implicit correction for flux-corrected transport. *Journal of Computational Physics* 71:1–20.

Pember, R.B., L.H. Howell, J.B. Bell, P. Colella, W.Y. Crutchfield, W.A. Fiveland, and J.P. Jesse. n.d. *An adaptive projection method for low-Mach number combustion*. Lawrence Berkeley National Laboratory report 41339. (submitted to *Combustion Science and Technology*).

Puckett, E.G. 1993. Vortex methods: An introduction and survey of selected research topics. In *Incompressible Computational Fluid Dynamics*, 335–408. New York: Cambridge University Press.

Pullin, D.I. 1992. Contour dynamics methods. *Annual Reviews of Fluid Mechanics* 24:89–115.

Ramamurti, R., R. Löhner, and W.C. Sandberg. 1994. *Evaluation of a scalable 3-D finite element incompressible flow solver*. American Institute for Aeronautics and Astronautics paper AIAA-94-0756. New York: AIAA.

Ramshaw, J.D., and J.A. Trapp. 1976. A numerical technique for low-speed homogeneous two-phase flow with sharp interfaces. *Journal of Computational Physics* 21:438–453.

Randles, P.W., and L.D. Libersky. 1996. Smoothed particle hydrodynamics: Some recent improvements and applications in meshless methods. *Computer Methods in Applied Mechanics and Engineering* 139:374–408.

Rehm, R.G., and H.R. Baum. 1978. The equations of motion for thermally driven, buoyant flows. *Journal of Research NBS/NIST* 83:297–308.

Rehm, R.G., H.R. Baum, and P.D. Darcy. 1982. Buoyant convection computed in a vorticity, stream-function formulation. *Journal of Research NBS/NIST* 87:165–185.

Richtmyer, R.D., and K.W. Morton. 1967. *Difference methods for initial-value problems*, New York: Interscience.

Roache, P.J. 1982. *Computational fluid dynamics*, Albuquerque, N. Mex.: Hermosa Publishers.

Roberts, K.V., and J.P. Christiansen. 1972. Topics in computational fluid mechanics. *Computer Physics Communications Supplement* 3:14–32.

Roe, P.L. 1981. Approximate Riemann solvers, parameter vectors, and difference schemes. *Journal of Computational Physics* 43:357–372.

———. 1985. Some contributions to the modelling of discontinuous flows. *Lectures in Applied Mathematics* 22:163–193.

———. 1986. Characteristic-based schemes for the Euler equations. *Annual Reviews of Fluid Mechanics* 18:337–365.

Rosenhead, L. 1931. The formation of vortices from a surface of discontinuity. *Procceedings of the Royal Society London* A134:170–192.

Sanders, R.H., and K.H. Prendergast. 1974. On the origin of the 3 kiloparsec arm to explosions in the galactic nucleus. *The Astrophysical Journal* 188:489–500.

Schmitt, H. 1994. Contour dynamics and the fast multipole method. *SIAM Journal of Scientific Computation* 15:997–1001.

Sod, G.A. 1978. A survey of several finite difference methods of systems of nonlinear hyperbolic conservation laws. *Journal of Computational Physics* 27:1–31.

Steger, J.L., and R.F. Warming. 1981. Flux-vector splitting of the inviscid gas dynamic equations with applications to finite-difference methods. *Journal of Computational Physics* 40:263–293.

Stellingwerf, R.F. 1991. Smooth particle hydrodynamics. In *Advances in the free-Lagrange method, lecture notes in physics* 395, eds. H.E. Trease, M.J. Fritts, and W.P. Crowley, 239–247. New York: Springer-Verlag.

Succi, S., E. Foti, and F. Higuera. 1989. Three-dimensional flows in complex geometries with the lattice Boltzmann method. *Europhysics Letters* 8:433–438.

Swegle, J.W., S.W. Attaway, M.W. Heinsein, F.J. Mello, and J.W. Hicks. 1994. *An analysis of smoothed particle hydrodynamics*. Sandia National Laboratory report SAND93-2513-UC-705, Sandia National Laboratories, Albuquerque, N. Mex.

Tannehill, J.C., D.A. Anderson, and R.H. Pletcher. 1997. *Computational fluid mechanics and heat transfer*. Washington, D.C.: Taylor and Francis.

Temam, R. 1969. On an approximate solution of the Navier-Stokes equations by the method of fractional steps. *Archive for Rational Mechanics and Analysis* 32: Part 1:135–153, Part 2:377–385.

Toro, E.F. 1997. *Riemann solvers and numerical methods for fluid dynamics*. Berlin: Springer.

Trease, H.E., M.J. Fritts, and W.P. Crowley, eds. 1991. *Advances in the free-Lagrange method, lecture notes in physics* 395. New York: Springer-Verlag.

Van Buskirk, R.D., and P.S. Marcus. 1994. Spectrally accurate contour dynamics. *Journal of Computational Physics* 115:302–318.

Van Leer, B. 1973. Towards the ultimate conservative difference scheme. I. The quest for monotonicity. In *Lecture Notes in Physics* 18, eds. H. Cabannes and R. Temam, 163–168. Berlin: Springer-Verlag.

———. 1979. Towards the ultimate conservative difference scheme. V.A second-order sequel to Godunov's method. *Journal of Computational Physics* 32:101–136.

———. 1982. Flux-vector splitting for the Euler equations. In *Eighth International Conference on Numerical Methods in Fluid Dynamics, lecture notes in physics* 170, ed. E. Krause. 507–512. New York: Springer-Verlag.

Warming, R.F., and R.M. Beam. 1977. On the construction and application of implicit factored schemes for conservation laws. In *Symposium on Computational Fluid Dynamics, SIAM-AMS proceedings* 11, 89–129, Philadelphia: SIAM.

Welch, J.E., F.H. Harlow, J.P. Shannon, and B.J. Daly. 1966. *The MAC method*. Los Alamos report LA-3425. Los Alamos, N. Mex.: National Laboratory.

Wen, Y., D.L. Hicks, and J.W. Swegle. 1994. *Stabilizing SPH with conservative smoothing*. Sandia National Laboratory report SAND94-1932-UC-705. Albuquerque, N. Mex.: Sandia National Laboratories.

Wilson, R.V., A.O. Demuren, and M. Carpenter. 1998. *Higher-order compact schemes for numerical simulation of incompressible flows*. NASA/CR-1998-206922. ICASE report no. 98-13. February 1998. NASA, Hampton, VA: Langley Research Center.

Woodward, P., and P. Colella. 1984. The numerical simulation of two-dimensional fluid flow with strong shocks. *J. Comp. Phys.* 54:115–173.

Xu, K., C.A. Kim, L. Martinelli, and A. Jameson. 1996. BGK-based schemes for the simulation of compressible flow. *International Journal for Computational Fluid Dynamics* 7:213–235.

Yee, H.C., and A. Harten. 1985. Implicit TVD schemes for hyperbolic conservation laws in curvilinear coordinates. In *AIAA 7th Computational Fluid Dynamics Conference*, 228–241. New York: AIAA.

Yee, H.C., R.F. Warming, and A. Harten. 1985. Implicit total variation diminishing (TVD) schemes for steady-state calculations. *Journal of Computational Physics* 57:327–360.

Zabusky, N.J., M.H. Hughes, and K.V. Roberts. 1979. Contour dynamics for the Euler equations in two dimensions. *Journal of Computational Physics* 30:96–106, reprinted in the *Journal of Computational Physics* 135:220–226, 1997.

Zabusky, N.J., and E.A. Overman. 1983. Regularization and contour dynamical algorithms I. Tangential regularization. *Journal of Computational Physics* 52:351–373.

10

Boundaries, Interfaces, and Implicit Algorithms

Boundary conditions are used to represent an infinitely large region of space using a finite computational domain, to describe boundary layers near walls, and to simulate details of chemical reactions, heat transfer, and other surface effects. Developing the correct boundary conditions to use in a numerical model involves complicated physical and numerical issues that make it relatively easy to make conceptual and programming errors. It is necessary to determine the correct boundary conditions to apply, how they should be implemented numerically, and whether there are inconsistencies between these boundary conditions and the description of the physical system within the computational domain. Although it is not always necessary to understand the solution completely to model the interior of a computational domain, implementing physically reasonable and sufficiently consistent boundary conditions requires a strong understanding of the interior and exterior phenomena and how they interact.

Interfaces are internal boundaries that have structure and can move with the flow. When interfaces are present, they greatly increase the complexity of the simulation. Additional physical processes that are not present within the flow field, such as surface tension, evaporation, condensation, or chemical reactions, can be important at these interfaces. Often the behavior of the interface has to be modeled phenomenologically as part of a much larger overall problem. This occurs when there are orders of magnitude difference in the gradients perpendicular to and parallel to the interface. For example, a shock may have a radius of curvature of centimeters or meters, but it may be only a micron thick. The layer in which ice melts and sublimates is only a fraction of a millimeter thick. Flame thicknesses may be on the order of centimeters, millimeters, or less. In these cases, the interface can often be treated as a discontinuity on macroscopic length scales.

This chapter also discusses algorithms for solving sparse matrix systems arising from finite-difference and finite-volume numerical analyses. We address the problem of how to solve certain types of systems of algebraic equations that arise naturally in the implicit and nonlocal solution methods described in previous chapters. These methods have been subjects of entire books. Here we present a short summary discussion with suggestions, and leave details to references.

10–1. Boundary Conditions

Boundary conditions can be an endlessly difficult issue as their selection and implementation is quite problem dependent. Relatively little of a general nature has been written about the edges of the computational domain compared to the volumes that have been written about the numerical techniques used in the interior. Nonetheless, more attention has necessarily been given to treating boundary conditions as numerical methods for treating the flow inside a computational domain have improved. This, at least in part, reflects the fact that more accurate solution methods for the interior of a computational domain usually require more accurate boundary conditions. In this section, we try to provide some general information, references, and rules of thumb that should help to select appropriate boundary conditions for a reactive-flow model.

10–1.1. General Comments on Boundary Conditions

All problems with an infinite domain must be mapped onto a finite domain for numerical computation. Since reducing the number of cells in the domain is an inexpensive way to reduce the cost of a simulation, the finite domain should be as small as possible. Thus the numerical model must represent an open or unconfined boundary with only a finite number of degrees of freedom. The challenge here is to keep the implementation of boundary conditions as simple as possible and yet avoid unphysical reflections and other effects that harm the important parts of the solution.

Actual physical confinement is usually provided by a wall or an obstacle. The confining surface may be inactive and so may be modeled by an apparently simple condition, such as constant temperature or adiabatic walls, on the numerical model at the boundary. Alternately, the wall material itself may play an active role in the chemistry and physics. In this case, considerably more complex models of boundary interactions are needed. We will distinguish surfaces as either *active surfaces* or *inactive surfaces*.

The types of boundary conditions required also depend on the types of differential equations that must be solved. For example, a one-dimensional, second-order differential equation may be written in the form

$$a\frac{\partial^2\rho}{\partial t^2} + b\frac{\partial^2\rho}{\partial t\,\partial x} + c\frac{\partial^2\rho}{\partial x^2} + d\frac{\partial\rho}{\partial t} + e\frac{\partial\rho}{\partial x} + f\rho + g = 0, \tag{10–1.1}$$

where ρ may represent one of the system variables, and the coefficients a, b, c, d, e, f, and g may be functions of position and time.

Depending on whether the discriminant $\mathcal{D} \equiv b^2 - 4ac$ is positive, zero, or negative, the equation is classified as hyperbolic, parabolic, or elliptic, as shown in Table 10.1. Because the coefficients can vary in time and space, the class of an equation may vary with time and location. When the coefficients are functions of the dependent variable ρ, the equation is nonlinear. These definitions may be generalized to multidimensions. For hyperbolic equations, the flow characteristics defined in Chapter 9 are real, and the speed of propagation of a signal is finite. In parabolic equations, such as diffusive equations, a propagation speed is not physically meaningful and the characteristics are complex. In elliptic equations, there is effectively an infinite propagation speed, so that the solution at any time is related to the solution on all the boundaries at the same time.

Table 10.1. Classification of Second-Order Partial Differential Equations*

Type	Condition*	Example	
Hyperbolic	$b^2 - 4ac > 0$	Wave equation	$\nabla^2\rho - \frac{1}{c}\frac{\partial^2\rho}{\partial t^2} = 0$
Parabolic	$b^2 - 4ac = 0$	Diffusion equation	$\nabla^2\rho - \frac{1}{\kappa}\frac{\partial\rho}{\partial t} = 0$
		Schrödinger equation	$\nabla^2\rho + \frac{2m}{\hbar^2}(E - V)\,\rho = 0$
Elliptic	$b^2 - 4ac < 0$	Laplace equation	$\nabla^2\rho = 0$
		Poisson equation	$\nabla^2\psi = \xi$

* Based on equation (10–1.1)

The treatment of boundary conditions necessarily differs greatly depending on the type of equation. In hyperbolic systems that simulate convection and acoustic phenomena, waves may travel much faster than the fluid flows through which they propagate. These waves carry information through the medium in all directions and thus information can enter the computational grid as well as leave it. In parabolic systems, the information flow is generally unidirectional, so the solution can be advanced preferentially in one direction. Diffusion fluxes carry the dependent variable in only one direction at each point. In certain supersonic, compressible flows, all of the characteristics move in one direction. Then the one-sided or direction-biased flow equations can be integrated along streamlines.

There are several ways to implement boundary conditions in numerical models:

1. Expand the continuum fluid variables in a linear superposition of basis functions with the boundary conditions built into each of them. Then any combination of basis functions automatically satisfies the boundary conditions. This approach cannot be applied systematically to most problems or to most numerical solution procedures.
2. Develop special finite-difference formulas for the boundary cells. These formulas have the general form of those used in the interior, but use auxiliary relations to replace grid variables that would lie outside the computational domain.
3. Develop extrapolation formulas from the interior to cells outside the computational domain that continue the numerical mesh a finite distance beyond the boundary. Boundary cells are then treated as interior cells. Formulas such as those developed for the previous method can be used to define the values in the outside cells.
4. Develop an analytic formulation for the boundary variables that uses information about the behavior of the system as it approaches infinity.

The complexity of the problem, the availability of fast computer memory, the specific numerical algorithms used, and personal preferences come into play when selecting one of these approaches.

Many implementations of boundary conditions require extra cells outside the computational domain. These cells are often called *guard cells* or *ghost cells*. Figure 10.1 shows a two-dimensional uniformly spaced grid where the boundary of the computational domain has been outlined by thicker lines. The grid has N_x cells in the x-direction, extending from x_L to x_R, and N_y cells in the y-direction, extending from y_B to y_T. The two rows of cells

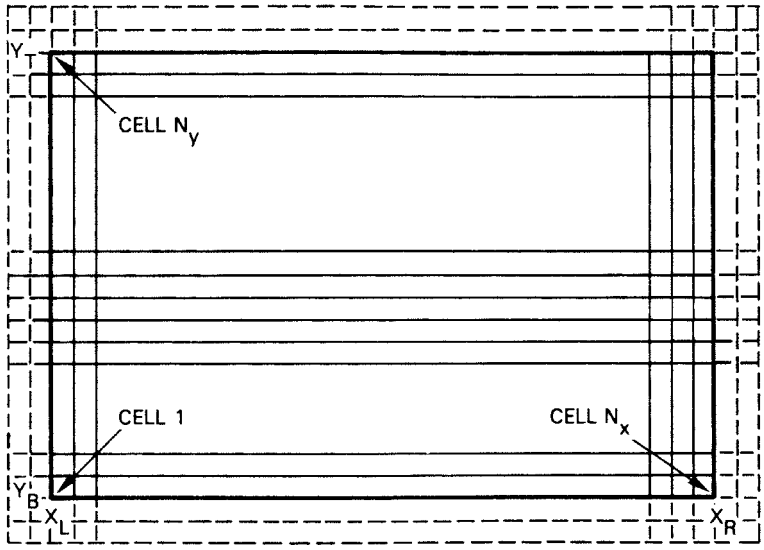

Figure 10.1. A two-dimensional computational domain with two rows of guard cells. Values of variables must be specifed in these cells to model various boundary conditions.

in dashed lines that surround the computational grid are the *guard cells* or *ghost cells*. By assigning appropriate values to the guard cells, the fluid elements inside the domain are forced to satisfy an imposed mathematical and (hopefully) physically consistent condition. Usually only one or two layers of guard cells are needed. When the values of variables assigned to guard cells are stored in the same arrays as the interior variables, the calculation can be advanced on the entire grid using a single set of finite-difference equations.

To reduce the amount of memory required in three-dimensional simulations, guard cells are sometimes used for only one plane at a time. The worst case is that of a uniformly spaced grid over the entire computational domain. For example, a $20 \times 20 \times 20$ grid uses 8,000 memory locations per variable. If this grid is extended two guard cells in all directions, the resulting $24 \times 24 \times 24$ grid has 13,824 cells, almost a factor of two increase with no added resolution in the internal flow! Because guard cells provide such a convenient approach to boundary conditions, it may be useful to have the additional computer memory available. With the increase in computational resources available in the last ten years and the development of technology for using adaptive and nonuniform meshes (see Chapter 6), memory limitations have become much less of a problem. In parallel computing, however, the problem reemerges because the surface area of small blocks is a much larger proportion of the total memory. These small blocks are the result of domain decomposition algorithms in parallel processing.

10–1.2. Boundary Conditions for Confined Domains

Ideal Symmetry or "Nonboundary" Conditions

The easiest way to treat boundary conditions accurately and consistently is to eliminate them. This can be done in regions where the solution satisfies a symmetry condition, as shown schematically in Figure 10.2. Boundary points can be treated as interior points

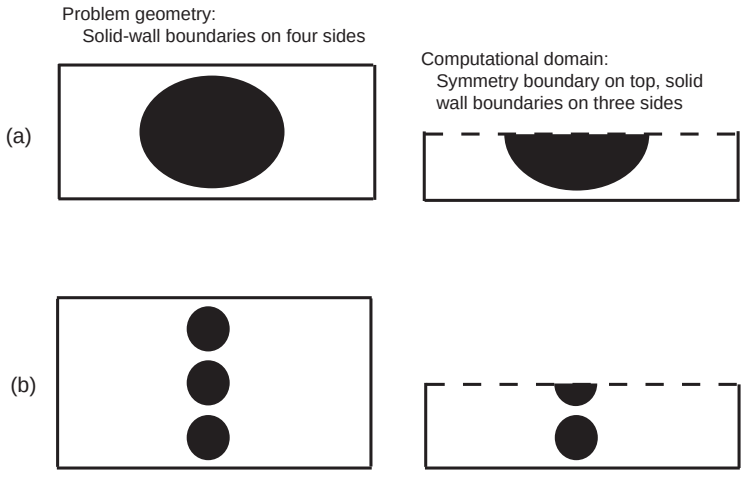

Figure 10.2. Schematic of two cases in which symmetry boundary conditions may be used. The computational grids for both the full domains (left figures) and half domains (right figures) require rows or columns of ghost cells on the tops and sides.

by adding guard cells whose values are known exactly from the symmetric values at interior locations. In these cases, guard cells simplify programming because only one set of difference formulas applies to all of the cells, whether or not they are near the boundary.

Symmetry conditions can often be applied in the interior of a system as well as at a boundary. A system may have a natural symmetry plane or line, such as the axis of an axisymmetric flow. Often a symmetry plane is a good approximation to solving a problem on a larger computational domain, as in the case of two equal and opposite impinging jets. Thus simple symmetry and reflection boundary approximations can be extremely useful for reducing the cost of a simulation. Note that when the size of the computed system is decreased relative to the physical system modeled, such as when symmetry boundary conditions are used, asymmetric physical effects that are on the scale of the size of the physical system may be excluded from the computational solution. This is problem dependent and occasionally very important.

Figure 10.3 shows the right boundary and last few cells of two one-dimensional grids. The edge of the grid is shaded to indicate a wall or to mirror an image system. Several guard cells are indicated beyond the boundary on the right. The upper figure depicts the symmetry boundary at the outer interface of the last cell. This is called a *cell-interface boundary*. The lower figure shows a *cell-centered boundary*, in which the symmetry plane passes through the center of the last cell. Similar figures could also be drawn for the left boundary and the first few cells of the grid. Figure 10.3 also shows two different interpretations of the discretization: a piecewise-constant representation in the top panel and a piecewise-linear representation in the bottom panel. Choosing between the four options shown in Figure 10.3 is generally a matter of taste, as modified by the requirement of numerical stability and the desire for simplicity.

When the grid locations are the cell centers, the piecewise-linear representation is the most natural to use because the discrete values of the fluid variables can be interpreted as the specific values at boundary locations. In this interpretation, the mass conservation

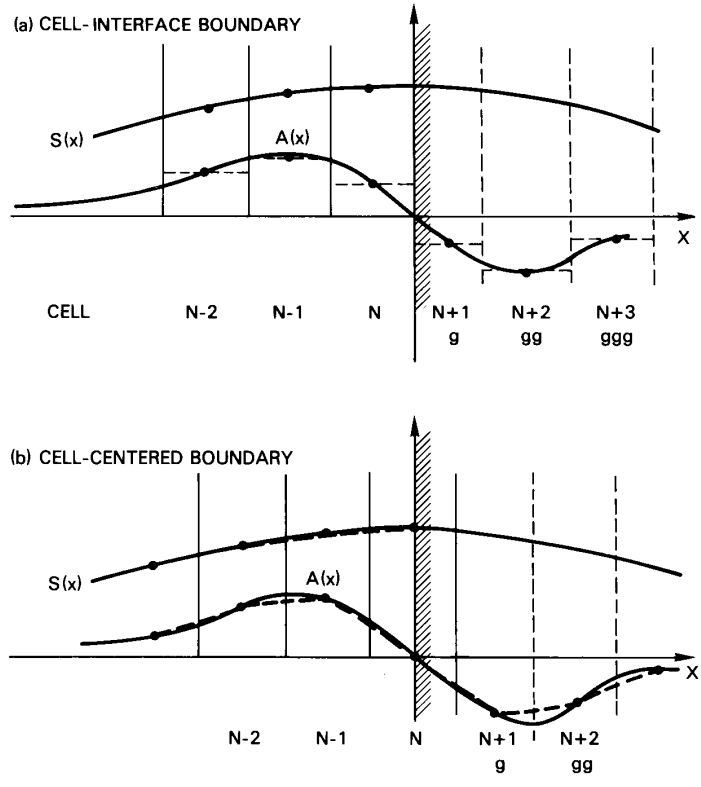

Figure 10.3. Different applications of symmetry boundary conditions with guard cells. The interior of the computational domain is on the left of the shaded band, and the guard cells are on the right. $S(x)$ is a symmetric function and $A(x)$ is an antisymmetric function. Both $S(x)$ and $A(x)$ have zero value on the x-axis. (a) Piecewise-constant representation with a cell-interface boundary. (b) Piecewise-linear representation with a cell-centered boundary.

sum described by equation (6–2.7) always has a different, half-cell term at the boundary. By placing the symmetry plane at the known cell-center location, there is no ambiguity about the location of the boundary.

When the grid locations are the cell interfaces, it is best to use the piecewise-constant interpretation. In this representation, the boundary and the interior cells are complete, so it is natural to end the computed domain at the edge rather than the middle of the last cell. Boundary fluxes enter and exit the system at interfaces where the geometric areas as well as the variable values must be specified. Conservation is ensured by controlling the fluxes at the cell interfaces.

Each panel in Figure 10.3 shows the two typical fluid variables, $S(x)$ and $A(x)$, where $S(x)$ is symmetric and $A(x)$ is antisymmetric with respect to the boundary plane. In addition to variables $A(x)$ and $S(x)$, there are also periodic functions, an example of which is $P(x)$. Periodic boundary conditions are variants of the symmetry conditions in which values at some interior location are used to set guard-cell values at locations that appear quite distant. For example, the values of the variables at cell $N + 1$ are equal to those at cell 1 in

Table 10.2. Guard-Cell Values*: Symmetric *(S)*, Antisymmetric *(A)*, Periodic *(P)*

Left Boundary	Right Boundary
Interface-Centered Boundaries	
$S_g = S_o = S_1$	$S_g = S_{N+1} = S_N$
$S_{gg} = S_{-1} = S_2$	$S_{gg} = S_{N+2} = S_{N-1}$
$A_g = A_o = -A_1$	$A_g = A_{N+1} = -A_N$
$A_{gg} = A_{-1} = -A_2$	$A_{gg} = A_{N+2} = -A_{N-1}$
$P_g = P_o = P_N$	$P_g = P_{N+1} = P_1$
$P_{gg} = P_{-1} = P_{N-1}$	$P_{gg} = P_{N+2} = P_2$
Cell-Centered Boundaries	
$S_g = S_o = S_2$	$S_g = S_{N+1} = S_{N-1}$
$S_{gg} = S_{-1} = S_3$	$S_{gg} = S_{N+2} = S_{N-2}$
$A_g = A_o = -A_2$	$A_g = A_{N+1} = -A_{N-1}$
$A_{gg} = A_{-1} = -A_3$	$A_{gg} = A_{N+2} = -A_{N-2}$
$A_1 = 0$	$A_N = 0$
$P_g = P_o = P_{N-1}$	$P_g = P_{N+1} = P_2$
$P_{gg} = P_{-1} = P_{N-2}$	$P_{gg} = P_{N+2} = P_3$
$P_1 = P_N$	$P_N = P_1$

* Subscript g indicates the first guard cell and gg indicates the second guard cell.

a periodic system. Periodic boundary conditions arise in circular systems such as stacks of turbine blades, cylindrical systems, or spherical systems. These cases are relatively straightforward to treat numerically.

We generally assume that the guard cells are the same size as the corresponding cells just inside the boundary. Table 10.2 lists simple guard-cell formulas for symmetric, antisymmetric, and periodic variables, for both interface-centered boundary systems and cell-centered boundary systems. The entries in this table show another difference in the two representations. With a cell-centered boundary, the two half-cell boundary elements have the same value. Thus there is a constraint on the interior values as well as the guard-cell values. In the case of the periodic function, the values of the variable just outside of the left boundary are the same as the values just inside on the right.

A slightly more general formulation that includes the symmetry, antisymmetry, and periodicity conditions is given by

$$f_g^n = bf_{sc}^n + c + pf_{pc}^n, \tag{10–1.2}$$

where f_g^n is the value of $f(x)$ at the guard cell at timestep n, f_{sc}^n is the current value of f at the symmetry cell (sc) point inside the domain, f_{pc}^n is the periodic cell value of f at timestep n, b and p are boundary-condition factors, and c is an additive constant. All of the cases in Table 10.2 can be obtained by setting $c = 0$ and making b and p appropriate combinations of $+1$, 0, and -1. Equation (10–1.2) also allows more complicated and realistic boundary conditions, including the inflow and outflow conditions described below.

Table 10.3. Free-Slip Flow along an Insulating Hard Wall*

Boundary at Cell Interface	Boundary at Cell Center
Density, Temperature, and Pressure:	
$\rho_g = \rho_s$	$\rho_g = \rho_s$
$T_g = T_s$	$T_g = T_s$
$P_g = P_s$	$P_g = P_s$
	ρ_{wall}, T_{wall}, P_{wall} calculated
Tangential Velocity:	
$v_{\parallel g} = v_{\parallel s}$	$v_{\parallel g} = v_{\parallel s}$
	v_{wall} calculated
Normal Velocity:	
$v_{\perp g} = -v_{\perp s}$	$v_{\perp g} = -v_{\perp s}$
	$v_{\perp \text{wall}} = 0$
Species Number Densities:	
$n_{i,g} = n_{i,s}$	$n_{i,g} = n_{i,s}$
	$n_{i,\text{wall}}$ calculated

* Subscript s indicates value taken at the symmetry cell.

A generalization of this equation, called the *uniform boundary algorithm*, includes the finite-volume formulas for the flow occurring inside the computation domain (Weber et al. 1993). Whether a cell is inside the domain or at the boundary is determined by a matrix of coefficients of various terms in the generalized equation.

When a flow with both normal and tangential components involves an inactive wall, the physical conditions in the guard cells can be found from the values in nearby interior cells. This is important for bounded Euler flows and for high-Reynolds-number viscous flows for which the viscous boundary layers are approximated by additional phenomenologies. Both symmetry and antisymmetry conditions must be applied for different variables at such a wall. The lowest-order condition is symmetric, as the density, temperature, and pressure have zero slope at the wall. The tangential velocity for *free-slip* conditions is symmetric, but the normal velocity is antisymmetric, guaranteeing a zero value at the wall. The guard-cell values of variables for flow involving a wall are given in Table 10.3.

Many values of the physical variables on the boundary cannot be determined by applying finite-difference equations with symmetry conditions directly. It is often necessary to develop modified boundary formulas that depend on the information from the interior. This allows representation of more physically complex situations such as viscous and turbulent boundary layers. The extrapolations generally use one-sided formulas.

Diffusion or Edge-Flux Boundary Conditions

Diffusion is one of the easiest of the nonlocal phenomena to simulate accurately (see Chapter 7). Similarly, formulating boundary conditions for diffusion problems is usually straightforward. For example, consider the finite-difference diffusion formula given in

Chapter 4 for a diffused quantity ρ,

$$\frac{\rho_j^n - \rho_j^{n-1}}{\Delta t} = \frac{D\theta}{(\Delta x)^2}\left[\rho_{j+1}^n - 2\rho_j^n + \rho_{j-1}^n\right] + \frac{D(1-\theta)}{(\Delta x)^2}\left[\rho_{j+1}^{n-1} - 2\rho_j^{n-1} + \rho_{j-1}^{n-1}\right], \tag{10–1.3}$$

where θ is the implicitness parameter. New values ρ_1^n and ρ_N^n must be specified in terms of fictitious values at guard cells. These values can be used to specify the gradient of ρ at the boundaries, and so to control the flux of ρ entering or leaving the computational domain during a timestep.

In some circumstances, the external guard-cell value, ρ_{N+1}^n, is known at all timesteps n. This situation occurs when a constant-temperature wall bounds a system described by an equation for thermal conduction. Then the known boundary values are used to advance the solution of equation (10-1.3) from step $n-1$ to step n.

Sometimes the flux of ρ is known, which means that the gradient of ρ is given at the boundary. In this case, the boundary value, ρ_N^n, and the guard-cell value, ρ_g^n, change consistently so that the gradient,

$$G^n = \frac{\rho_{N+1}^n - \rho_N^n}{\Delta x}, \tag{10–1.4}$$

is fixed and ρ_{N+1}^n can be eliminated from equation (10–1.3) using equation (10–1.4).

Boundary Layers and Surface Phenomenologies

A solid-walled container filled with gas may be chemically inert, thermally insulating, absolutely rigid, and perfectly smooth. If these approximations are acceptable, the simplified symmetry and guard-cell algorithms described above can be used. Boundaries often play an active role, such as might occur at interfaces separating different phases or materials. Then the topic of boundary conditions may have to be expanded to include subgrid phenomenologies for representing other types of physics not generally resolved in the simulations. For example, surface phenomenologies can be used to model catalytic reactions and thermal or viscous boundary layers at walls, or physical processes such as condensation, evaporation and ablation. In all of these cases, the physics of these processes is not actually resolved because this would require including more complex physical effects or much more resolution at the interface.

Therefore, simple, underresolved models of surface phenomenology must at least satisfy conservation equations at and through the boundaries. This means that fluxes of mass, momentum, energy, and enthalpy, which enter or leave the computational domain through the boundaries, must exactly equal the fluxes to the exterior world. Causality and the conservation laws provide valuable constraints on the processes and the models developed to represent them.

Phenomenological models that use relatively large computational cells have been developed to describe viscous boundary layers, whose effects derive from relatively small-scale phenomena. The usual macroscopic treatment of the Navier-Stokes equations assumes the *zero-slip condition*, which means that the tangential and normal velocity components at

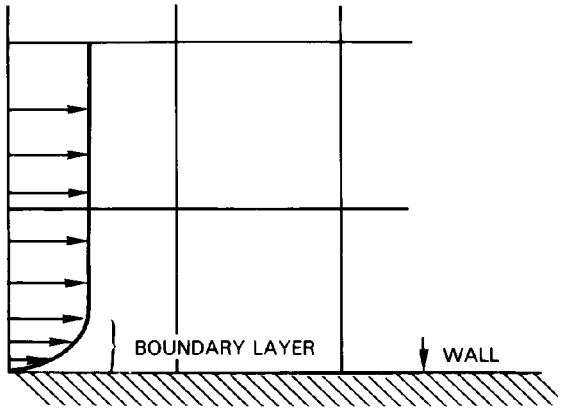

Figure 10.4. Schematic of a grid in which the physical boundary layer is substantially smaller than the computational cell adjacent to the wall.

the wall are both zero. On a microscopic scale, molecules rebounding from a rigid wall are assumed to lose all memory of their collective flow properties before they collide with the wall. They will then have equal probability of scattering forward or backward relative to the flow of the fluid far from the wall. The result is that a thin laminar boundary layer forms near the wall, as shown in Figure 10.4. Because this boundary layer develops on a spatial scale that is expensive to resolve in standard fluid computations, a subgrid phenomenology is often used to satisfy the physical conditions at this rigid surface.

Consider a numerical model of a catalytic surface. The boundary condition at this surface must estimate the amount of the reactant that encounters the boundary and evaluate the probabilities of each possible reaction pathway. If the computational cells at the walls are large, the reacting species appear to be spread throughout nearby cells, even though they actually may be concentrated much closer to the wall. If subsequent reactions depend on the volumetric density of these reactants, the model underestimates the volumetric rates at the wall and greatly overestimates them for the distance represented by a few cells away from the wall. Surface effects appear numerically as volume averages leading to spurious rates. The problem disappears when the grid is fine enough to resolve molecular diffusion in the vicinity of the walls.

10–1.3. Boundary Conditions for Unconfined Domains

Simulating an unconfined flow requires representing an effectively infinite region as a finite computational domain using what are, in effect, adjustable boundary conditions with only a finite number of degrees of freedom. In this case, the boundary conditions must transmit information to and from the entire outside world, and they must properly absorb, and not reflect, signals generated inside the computational domain. Gustafsson and Kreiss (1979) have shown that systems coupled to an exterior domain can be rigorously computed on a bounded domain only when the variables and coefficients in the problem become constant at infinity. When such simplifying conditions are not met, the accuracy limits of the approximations used involve issues that go beyond those of the solution methods used to compute the behavior inside the domain.

Two main approaches have been used. In the first, the infinite region is mapped into a finite domain by analytically redefining the independent spatial variables. The major problem with this approach is that spatial resolution in the region of interest becomes inadequate to propagate information at the wavelengths of interest (see, for example, Grosch and Orszag [1977]). The second approach analytically models the influence of the exterior domain on the finite boundaries. The problem here is that the shorter-wavelength components of the variables can now propagate up to the boundaries, but they are partially reflected in an unphysical way when an exact analytic condition is not available. Section 10–2 considers recent treatments for high-order boundary conditions.

Often a combination of these approaches is the best choice. First, the cells should be made progressively larger away from the central region of interest. This type of gridding is shown in Chapter 6 in Figures 6.7, 6.9, and 6.11–6.18. The boundary of the computational domain is viewed as the frame of a window outlining a portion of an infinite medium. Expanding the cells near the boundary pushes boundary-condition problems farther away from the interior. Errors still arise from lack of knowledge of the solution in the exterior region, but these have less of an effect, and so affect the solution only after the delay that it takes for the influence of the numerical boundary conditions to reach the central region. Analytic or phenomenological models are used to approximate the values of the variables in the region outside of the computed domain. This is done using computed values near the edge of the domain and auxiliary information about the presumed exterior behavior of the solution. In this way, potentially unstable numerical extrapolations beyond the computational domain are replaced by more stable interpolations to approximate external functions.

In many cases, the behavior of the physical system is closely coupled to the external world. This coupling often appears as conserved integrals over the discretized domain which are governed by interactions with parts of the outside fluid. Sometimes the value of an important physical property of the interior system is known *a priori*, rather than being determined by the simulation. For example, a flame in open air burns essentially at constant pressure, so the pressure at infinity becomes a constraint on the dynamics of the flow. The rate of expansion in a small burning element of a premixed gas is governed by the rate of exothermic chemical reactions. The density decreases as fast as the temperature increases to keep pressure essentially constant. There is a constraint on the fluxes into or out of the computational domain as a result. This constraint represents feedback from the outside world that must be included if the answers are to be quantitatively correct.

In unconfined systems, boundary conditions for waves and convection are more difficult than for diffusion. Because waves are often weakly damped as they propagate, smoothing from diffusion cannot be relied on to mask errors at boundaries in the same way it does for diffusive phenomena. The most difficult boundary conditions have always been continuitive boundary conditions used to describe a fluid flowing into and out of a computational mesh with a superimposed wave field. The complexities of the fluid motion mean that localized vortex structures cross the boundary. Thus the coefficients in the equations are time varying, and no analytic solution exists.

Continuitive or Inflow-Outflow Boundary Conditions

Chapter 9 discussed the *characteristics* of a flow. These may be used to determine how much information can be specified independently at a boundary for given flow situations.

In all but the simplest linear problems, the algorithms for boundary conditions can only be approximate because inward and outward propagating waves and pulses become locally indistinguishable due to the nonlinear behavior of a fluid. By separating the fluid disturbance into its constituent flow characteristics (see Section 9–1.2) and extrapolating each of these out to the guard cells, more stable, though still approximate, outflow conditions are obtained for any but the most finicky convection algorithm.

Use of characteristics and wave systems to formulate boundary conditions has been discussed by a number of authors, and is considered in Section 10–2. In particular, we recommend the articles by Pulliam (1982), Oliger and Sundström (1978), Yee (1981), Thompson (1987, 1990), Givoli (1991), Poinsot and Lele (1992), Grinstein (1994), and Tam (1998). The incremental changes in the evolution of a fluid system can be separated locally into a number of well-defined characteristic behaviors, generally in terms of the normal modes of the linearized system.

Table 10.4 describes simple, approximate boundary conditions for a fluid flowing off the edge of a finite computational domain. These formulas for the guard-cell values take into account the continuity of flow in the vicinity of the boundary. They give low-order "interpolation" formulas for guard-cell values assuming the flow behavior at infinity is given. The lowest-order extrapolation for guard-cell values uses the adjacent cell values, that is, $b = 1$ and $\tau = \infty$. The next higher-order extrapolation uses the two cells just inside the boundary to extrapolate linearly to the guard cells. The values of the variables at or near infinity, such as ρ_∞, feed information about the external domain into the computational domain. Suitable definitions of the coefficients in the formulas of Table 10.4 can be used to implement approximate characteristic boundary conditions. Without these terms, that is,

Table 10.4. Interpolation of Flow off a Grid Boundary

Boundary at Cell Interface or Cell Center*

Density, temperature, and pressure:

$\rho_g = b\rho_s + c_\rho$,

where $b = \left[1 - \frac{\Delta t}{\tau}\right]$,

$c_\rho = \frac{\Delta t}{\tau}\rho_\infty$,

τ is time constant for the relaxation of boundary conditions to the far field value.

$T_g = bT_s + c_T$

$P_g = bP_s + c_P$

Different values of τ for temperature, density, and pressure may be appropriate for subsonic flows.

Tangential velocity:

$v_{\parallel g} = v_{\parallel s}$

Normal velocity:

$v_{\perp g} = v_\perp$ (same sign to extrapolate off the boundary)

Species number densities:

$n_{i,g} = bn_{i,s} + c_{n_i}$

* Subscript g refers to the guard cell and subscript s refers to the symmetry cell inside the computational domain for the particular type of grid chosen.

with $\tau = \infty$, a simulation cannot relax to the pressure at infinity, and this leads to growing errors called *secular errors*.

Radiative or Wave-Transmission Boundary Conditions

Consider a situation in which there is essentially no convective flow at the boundaries of the computational domain, but there are waves that propagate out of the domain into the exterior region. The boundary conditions must simulate this wavelike radiation. If the radiation conditions are specified incorrectly, waves are mistakenly reflected at the edge of the domain and appear to the simulation as energy propagating in from infinity. This could lead to inaccurate, incorrect, or confused conclusions.

As an efficient but not fully general approach, an artificial dissipative region may be set up in the few layers of cells adjacent to the boundary. When fluid flow is also involved, this dissipation region is often called a Rayleigh viscous zone. As a wave disturbance propagates into this region, it encounters a medium of progressively greater damping and becomes trapped and dissipated with minimal reflection. Because there is some signal reflection off the gradient, the optimal absorbing boundary is obtained when the spatial variation of the boundary damping coefficient is gradual. Then the waves become "trapped" before the unphysically reflected wave component becomes appreciable.

This damping technique has been used for electromagnetic waves, sound waves, and gravity waves, although it does not work for bulk convection. It can be added simply to a model generated by more idealized methods. Different wavelengths may be treated differently by this approach, but the maximum damping can usually be arranged to coincide with the strongest outgoing wavelengths. Although the absorption is not perfect, four or five cells are often enough to reduce the amplitude of the spuriously reflected wave by more than an order of magnitude. Ten cells should be enough to guarantee 98 to 99 percent absorption.

In linear problems, it is sometimes possible to describe the incoming and outgoing wave systems at a boundary by an analytic expression. When this can be done, nonreflective boundary algorithms, sometimes called *radiative boundary conditions*, can be developed by eliminating the incoming wave system and analytically extrapolating the behavior of the outgoing waves. Various types of functional expansions can be used to describe the outer solutions. This process is very different from the local analysis of characteristics discussed above. Usually, to determine the wave components, a Fourier expansion, or some equivalent nonlocal analysis, is required along the entire boundary for two or three layers of cells. The outer solutions must be matched to the inner solutions at the boundaries of the domain. For example, spherical harmonics may be matched at a constant radius to outgoing solutions inside this radius. In cylindrical coordinates, a circle can be matched with Bessel-function solutions for outgoing waves. In ducts or semi-infinite geometries, Cartesian plane waves can be used.

In almost all cases, the far-field variables are linearized so the individual components of the outwardly propagating radiation can be combined additively. Once the functional form of the expansion is known, we can derive differential relations, and these relations are then used as boundary conditions to match the computed solution to the asymptotic expansion valid near infinity. These radiative boundary conditions become increasingly accurate as they match the solution to more terms in the exterior expansion and as the

boundaries are taken far enough from the source of waves so that the amplitudes become small. Conversely, they are limited by nonlinearity. An example of a nonlinear problem is a pocket of hot gas leaving the computational domain and changing the speed of sound for acoustic-wave radiation conditions.

A different approach is proposed by Engquist and Majda (1977, 1979), who constructed a pseudodifferential operator to exactly annihilate outgoing waves. This pseudodifferential operator is a global boundary operator. In order to derive local boundary conditions, they expanded the pseudodifferential operators in the deviation of the wave direction from some preferred direction of propagation. In this manner they construct local boundary conditions to absorb waves in a progressively larger band around a given propagation direction.

10–1.4. Conditions for a Thermal Boundary Layer

Determining phenomenological boundary conditions to represent a thermal boundary layer is a challenging problem that occurs in some reactive flows and is usually not treated very accurately. Here we use this problem as an example of coupling a boundary phenomenology to a model to simulate complicated phenomena relatively inexpensively.

Consider a shock in a chemically reacting gas medium that reflects from a smooth, cold metal wall. Shortly after the shock reflects, the fluid velocity near the wall returns essentially to zero. The shock heats the gas, and compresses it adjacent to the wall. The temperature in the wall begins to rise immediately and the gas temperature adjacent to the wall begins to drop. All the energy taken out of the fluid enters the wall, first as a thin boundary layer and later in a broad, diffuse profile. Because the heat capacity is much lower for the gas than for the metal, the gas undergoes a larger temperature change than the metal.

We are primarily interested in boundary conditions for the gas, not the details of the temperature in the metal as a function of time and depth. To formulate a suitable model of the temperature in the metal from knowledge of its time evolution at the surface, several assumptions are necessary. Here we assume that the wall is rigid, and its thermal conductivity and heat capacity are finite and given.

Figure 10.5 shows the temperature profile in the vicinity of the wall at two times after the shock has reflected. The exposed surface of the wall is at $x = 0$, and the temperature there is $T_W(t)$. The fluid temperatures calculated in the two cells adjacent to the wall are T_N and T_{N-1}. These values are calculated at discrete timesteps and must be coupled to the changing wall temperatures. At later times, a rarefaction wave may reach the wall, reducing the gas temperature below the peak temperature of the wall. This leaves a complicated temperature profile in the wall, as shown in Figure 10.5b. The energy flux out of the wall reheats the gas, as indicated by the positive slope of the temperature profiles for T_N and T_{N-1}.

The correct temperature solution in the wall is a complicated, time-dependent profile that requires a detailed computation to know accurately. The first step in constructing a suitable phenomenology is to assume a shape for the spatial profile of the temperature in the wall with a few free parameters adjusted to ensure that energy is conserved and conducted out of the fluid as fast as it is conducted into the wall. Assuming a Gaussian profile for the shape, as shown in Figure 10.5, the free parameters are: $x_m(t)$, the distance

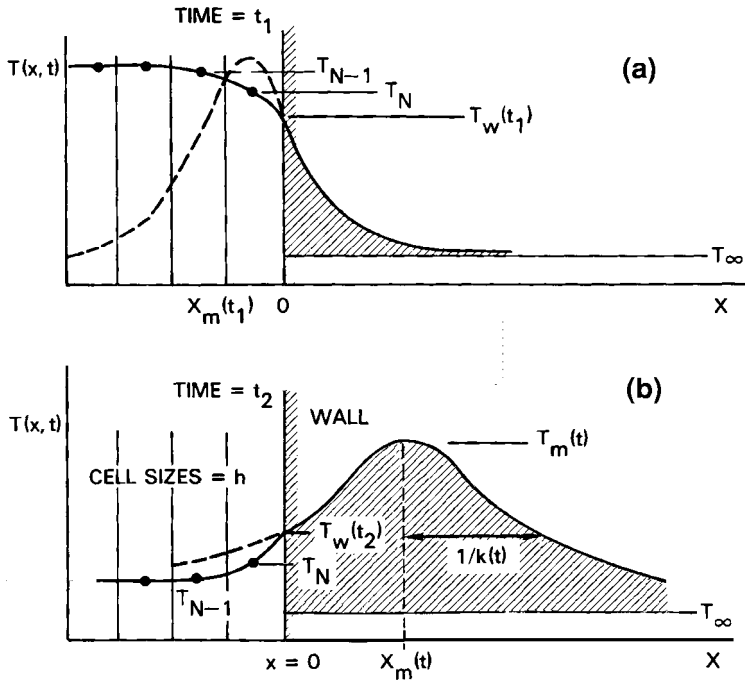

Figure 10.5. Temperature profile shown after a strong shock has reflected from a cold metal wall. (a) Time t_1 after shock reflection. (b) Time t_2 when a more complex profile evolves after a rarefaction reaches the wall.

from the surface to the peak of the Gaussian; $k(t)$, the characteristic inverse scale length of the Gaussian; $T_W(t)$, the temperature at the wall; and T_∞, the wall temperature far from the surface.

The assumed functional form of the temperature in the wall is then

$$T(x,t) = T_\infty + T_m(t)e^{-k^2(t)(x-x_m(t))^2}, \tag{10–1.5}$$

where

$$T_m(t) = (T_W(t) - T_\infty)e^{\eta^2(t)}, \tag{10–1.6}$$

with

$$\eta(t) \equiv -k(t)x_m(t). \tag{10–1.7}$$

Here $\eta(t)$ is positive when $T_N > T_W$, that is, when the peak of the Gaussian would be out in the fluid. When η is negative as shown in Figure 10.5b, the peak of the Gaussian is inside the wall.

The equations satisfied by the temperature inside the wall and inside the fluid are

$$\frac{\partial T}{\partial t} = \lambda_W \frac{\partial^2 T(x,t)}{\partial x^2} \qquad \text{for } x > 0, \tag{10–1.8a}$$

$$\frac{\partial T}{\partial t} = \lambda_N \frac{\partial^2 T(x,t)}{\partial x^2} + \dot{T} \qquad \text{for } x < 0. \tag{10–1.8b}$$

Here $\dot{T}$ indicates changes in the temperature of the fluid due to processes in the fluid itself, such as convection and chemical energy release. We have subsumed the heat capacities into the thermal-conduction coefficients, λ_N and λ_W (which in cgs units have units of cm^2/s).

Energy conservation inside the wall relates $T_W(t)$ to $k(t)$. The integrated energy entering the wall from the fluid is

$$E(t) = \int_0^\infty (T(x', t) - T_\infty) dx' = \frac{T_m(t)\mathcal{E}(\eta)}{k(t)}, \tag{10–1.9}$$

where $\mathcal{E}(\eta)$, related to the error function, is given by

$$\mathcal{E}(\eta) = \int_0^\infty e^{-(\eta'+\eta)^2} d\eta'. \tag{10–1.10}$$

The second relation connecting T_W, k, and η is derived by equating the fluxes used in equation (10–1.8) at the interface,

$$\lambda_N \frac{\partial T_N}{\partial x}\bigg|_N = \lambda_W \frac{\partial T}{\partial x}\bigg|_W. \tag{10–1.11}$$

In the fluid we write a finite-difference approximation for the left side of equation (10–1.11) and use the analytic form, equation (10–1.5), to evaluate the right side. The result is

$$\frac{T_W - T_N}{T_W - T_\infty} = -\left(\frac{\lambda_W \eta}{\lambda_N}\right) kh. \tag{10–1.12}$$

When constructing such computational models, it is important that they should be made to reproduce as many limiting analytic solutions as is reasonably possible. When the initial conditions are a half Gaussian in the wall with $\eta = 0$, and the fluid temperature in the gas varies in a way that keeps the temperature gradient at the wall zero, the solution inside the wall should evolve like the correct Gaussian decay profile. In this case the behavior of the inverse scale length $k(t)$ is known,

$$k(t) = \sqrt{4\lambda_W (t - t_o)}. \tag{10–1.13}$$

The adjustable parameter t_o is the time when the Gaussian was singular. In a general case, we try to choose an average, representative value for $k(t)$ that reduces to the desired result, equation (10–1.13), when all the energy is deposited at once. One way to do this is to choose a suitable value for t_o which averages over the time-varying energy input, and is biased in favor of the most recent changes in total energy. For example, one choice is

$$t_o(t) = \frac{\int_{-\infty}^t t' \left|\frac{dE(t')}{dt}\right|^4 e^{t'/\tau} dt'}{\int_{-\infty}^t \left|\frac{dE(t')}{dt}\right|^4 e^{t'/\tau} dt'}. \tag{10–1.14}$$

It is straightforward to verify that t_o is the correct constant when the energy deposition is a delta function.

There is another limiting case for which we know how the phenomenology should behave. When the energy input from the fluid to the wall is exponential, growing as

$$\left|\frac{dE(t)}{dt}\right| \approx e^{\Gamma t}, \tag{10–1.15}$$

the temperature profile in the wall should become exponential with a scale length k that does not change in time. This scale length is related to the exponential energy addition rate Γ by

$$k = \sqrt{\Gamma/\lambda_W}. \tag{10–1.16}$$

Using the fourth power of the energy-addition rate in equation (10–1.14) ensures that this scale length is recovered when the integrals are evaluated. The answer obtained for this scale length is

$$k = \sqrt{\left(\Gamma + \frac{1}{4}\tau\right)\Big/\lambda_W}, \tag{10–1.17}$$

which agrees with the theoretical result when the transient heating is fast, compared with the long memory time, τ. The adjustable time τ is included in equation (10–1.17) to bias the determination of t_o in the calculation of $k(t)$ to energy addition occurring in the recent past.

As the transient heating rate increases, we expect the exponential solution to be valid and indeed the model becomes quite accurate. When the transient heating decreases, the exponential solution is less valid and the model changes smoothly to an averaged decaying Gaussian that again evolves in the analytically prescribed manner.

The model just described is only a phenomenology with adjustable parameters, even though physical constraints (conservation, continuity, and two limiting cases) are built in. Further improvements are possible. The value of τ chosen for the simulation should be comparable to a memory time expected of the interior thermal solution. To implement this model, the fluid cell value T_N can be advanced from t to $t + \Delta t$ using equation (10–1.12) with all of the variables solved implicitly. The energy is advanced by calculating the flux that leaves the fluid and enters the wall and by changing the accumulated energy integral $E(t)$ by the correct amount. Then using equation (10–1.9) and evaluating equation (10–1.14), all the parameters can be determined at the new time, $t + \Delta t$. The integrals in equation (10–1.14) are accumulated at each timestep from the energy exchanged between the fluid and the wall. The implicit-solution approach requires an iteration to solve the nonlinear transcendental equations that arise. This is not a problem because it is done only at a boundary, involves only a few variables, and has much less stringent stability conditions than the numerical model used for the fluid.

10–2. Boundary Conditions for High-Order Algorithms

Since the last edition of this book, there has been growing recognition of the importance of high-order and monotone methods for solving the Euler and Navier-Stokes equations.

Table 10.5. Mathematical Boundary Conditions for Well-Posed Three-Dimensional Solutions

Boundary Type	Euler Equations	Navier-Stokes Equations
Supersonic inflow	5	5
Subsonic inflow	4	5
Supersonic outflow	0	4
Subsonic outflow	1	4

With the widespread use of these algorithms comes the need for correspondingly better boundary conditions. Part of this need arises because the high-order methods are less diffusive, so that some of the numerical diffusion that damped problems at boundaries is no longer present. But there are also issues that go beyond specific methods and are related to the general properties of partial differential equations and how they are discretized.

A considerable amount of work has been done to determine the correct mathematical boundary conditions required for obtaining well-posed solutions of the Euler and Navier-Stokes equations. Such conditions have been derived for the Euler equations (Kreiss 1970; Higden 1986; Engquist and Majda 1977, 1979; Gustafsson and Oliger 1982). The problem is more complex for the Navier-Stokes equations, for which analyses of what gives a well-posed solution has only been done for certain simple cases (Gustafsson and Sundström 1978; Oliger and Sundström 1978; Dutt 1988). The theory of characteristics provides some guidelines for the numbers of mathematical boundary conditions for Euler and Navier-Stokes flows, depending on whether the flow is inflow, outflow, subsonic, or supersonic. Table 10.5 gives these numbers (see, for example Poinsot and Lele [1992] or Grinstein [1994]). For the Euler equations, these numbers are suggested by the theoretical analyses referenced above. For the Navier-Stokes equations, it is necessary to add additional conditions for the diffusive transport effects (viscosity and thermal diffusion).

In compressible gas dynamics, there are typically three characteristics in one dimension: the entropy mode, which describes convection of density and temperature variations, and the right- and left-moving acoustic modes. By separating the solution near the boundary into its characteristics, those parts of the solution moving off the grid can be extrapolated from the interior, and those parts moving onto the grid can be specified externally. In an incompressible flow, the entropy mode remains, and vorticity becomes another conserved characteristic quantity.

Knowing the mathematically correct boundary conditions only solves one part of the problem. When the Euler or Navier-Stokes equations are discretized for computation, more is required than knowing the mathematical conditions that ensure that the solution is well posed. When the number of physical boundary conditions is less than the number of primitive variables, it is sometimes necessary to add other conditions. This is the case, for example, for an outflow boundary condition. The opposite problem may occur at a wall, where there could be too many equations and not enough unknowns, so that the problem is overconstrained. Therefore, even when the discretization is of relatively high order, the mathematical and discretized sets of equations do not solve the same exact problem.

In addition, the resulting numerical equations may be of higher order than the original equations. Thus the numerical solutions may include spurious numerical solutions that are not closely related to solutions of the original equations.

The result is that additional boundary conditions may have to be added to the first set. These additional constraints are the *numerical boundary conditions* whose function is to provide the additional information that the numerical solution requires, and to ensure compatibility between the numerical method and the mathematical formulation. The numerical boundary conditions complete the specifications required by the discretization and are there to minimize the amplitude of the extraneous solutions. The results of the simulations depend on the choice of both types of boundary conditions.

As described earlier, numerical boundary conditions are needed whenever there is no explicit boundary condition that fixes one of the dependent variables, but when the numerical implementation requires that some information be specified about this variable (Yee 1981). There are different approaches to specifying the numerical boundary conditions. Some of these are justified *a posteriori*; others have been determined by trial and error. Extrapolation of the value in the interior of the domain is common, such as extrapolating the pressure from the interior. This imposes a zero pressure gradient, which may cause problems because it is unphysical. Another approach is to use the conservation equations themselves to solve for the conditions that could be used on the boundaries.

There are now considerable, ongoing efforts to develop good sets of numerical boundary conditions for the Euler equations (Thompson 1987, 1990) and the Navier-Stokes equations (Poinsot and Lele 1992). In addition, this work has been extended to include detailed considerations of boundary conditions appropriate for sound waves transmitted and reflected from boundaries for aeroacoustics problems by Tam (1998), for thermoacoustics problems by Farouk, Oran, and Fusegi (2000), and for multispecies, highly exothermic flows by Ott (1999). The bottom line is that there are a number of good starting places for developing high-order boundary conditions when they are needed. These require a careful derivation and test for each new physical problem, as there is not yet a single consistent set of rules that works for all Euler, Navier-Stokes, reacting, and nonreacting problems.

10–2.1. Use of Characteristics

One approach that has proved useful for the Navier-Stokes equation is based on an analysis of the one-dimensional flow characteristics of the Euler equations at the boundaries. There is no exact definition of the characteristics for multidimensional systems, or even for one-dimensional Navier-Stokes systems. The approach is to infer the values of the wave amplitudes appropriate for the Navier-Stokes multidimensional case by examining a related, local, one-dimensional problem. The philosophy of this procedure, which provides the starting points for more complex analyses, is summarized below.

First, we need to write a set of three-dimensional nonreacting Navier-Stokes equations (see Chapter 2) at each boundary of the computational domain. For example, denote the three directions by $\mathbf{r} = (x_1, x_2, x_3)$, and the corresponding fluid velocities are

$\mathbf{v} = (v_1, v_2, v_3)$. Consider the equations at the x_1 domain boundaries (Poinsot and Lele 1992)

$$\frac{\partial \rho}{\partial t} + d_1 + \frac{\partial}{\partial x_2}\rho v_2 + \frac{\partial}{\partial x_3}\rho v_3 = 0 \tag{10–2.1}$$

$$\frac{\partial \rho E}{\partial t} + \frac{1}{2}v_j v_j d_1 + \frac{d_2}{\gamma - 1} + \rho v_1 d_3 + \rho v_2 d_4 + \rho v_3 d_5 + \frac{\partial}{\partial x_2}((\rho E + p)v_2) + \frac{\partial}{\partial x_3}((\rho E + p)v_3) = \frac{\partial}{\partial x_i}(v_j \tau_{ij}) - \frac{\partial q_i}{\partial x_i} \tag{10–2.2}$$

$$\frac{\partial \rho v_1}{\partial t} + v_1 d_1 + \rho d_3 + \frac{\partial}{\partial x_2}\rho v_1 v_2 + \frac{\partial}{\partial x_3}\rho v_1 v_3 = \frac{\partial \tau_{1j}}{\partial x_j} \tag{10–2.3}$$

$$\frac{\partial \rho v_2}{\partial t} + v_2 d_1 + \rho d_4 + \frac{\partial}{\partial x_2}\rho v_2 v_2 + \frac{\partial}{\partial x_3}\rho v_2 v_3 + \frac{\partial p}{\partial x_2} = \frac{\partial \tau_{2j}}{\partial x_j} \tag{10–2.4}$$

$$\frac{\partial \rho v_3}{\partial t} + v_3 d_1 + \rho d_5 + \frac{\partial}{\partial x_2}\rho v_3 v_2 + \frac{\partial}{\partial x_3}\rho v_3 v_3 + \frac{\partial p}{\partial x_3} = \frac{\partial \tau_{3j}}{\partial x_j}, \tag{10–2.5}$$

where we have used the summation convention over repeated indices i and j.

The various derivatives that are tangential to the boundary have been written explicitly. The derivatives normal to the boundary are written here as the set of vector elements $\{d_i\}$, which can be expressed as (Thompson 1987)

$$\mathbf{d} = \begin{pmatrix} d_1 \\ d_2 \\ d_3 \\ d_4 \\ d_5 \end{pmatrix} = \begin{bmatrix} \frac{1}{c_s^2}\left[\mathcal{L}_2 + \frac{1}{2}(\mathcal{L}_5 + \mathcal{L}_1)\right] \\ \frac{1}{2}(\mathcal{L}_5 + \mathcal{L}_1) \\ \frac{1}{2\rho c_s}(\mathcal{L}_5 - \mathcal{L}_1) \\ \mathcal{L}_3 \\ \mathcal{L}_4 \end{bmatrix} = \begin{bmatrix} \frac{\partial}{\partial x_1}\rho\, v_1 \\ \frac{\partial}{\partial x_1} c_x^2 \rho\, v_1 + (1-\gamma)\,\mu\, \frac{\partial p}{\partial x_1} \\ v_1 \frac{\partial v_1}{\partial x_1} + \frac{1}{\rho}\,\frac{\partial p}{\partial x_1} \\ v_1 \frac{\partial v_2}{\partial x_1} \\ v_1 \frac{\partial v_3}{\partial x_1} \end{bmatrix} \tag{10–2.6}$$

where c_s is the speed of sound ($c_s{}^2 = \gamma p/\rho$) and μ is the viscosity coefficient defined in Chapter 2 (see Table 2.4). The $\{\mathcal{L}_i\}$ are the amplitudes of the characteristic waves associated with characteristic velocities $\{\lambda_i\}$ for subsonic flows,

$$\lambda_1 = v_1 - c_s \tag{10–2.7}$$

$$\lambda_2 = \lambda_3 = \lambda_4 = v_1 \tag{10–2.8}$$

$$\lambda_5 = v_1 + c_s \tag{10–2.9}$$

Here λ_1 and λ_5 are the velocities of sound waves moving in the negative and positive x_1 direction, λ_2 is the convection velocity (the speed of entropy waves), and λ_3 and λ_4 are the velocities at which v_2 and v_3 are advected in the x_1 direction. The $\{\mathcal{L}_i\}$'s can be found

from the vector **d**, defined in equation (10–2.6),

$$\mathcal{L}_1 = \lambda_1\left(\frac{\partial p}{\partial x_1} - \rho\, c_s \frac{\partial v_1}{\partial x_1}\right) \tag{10–2.10}$$

$$\mathcal{L}_2 = \lambda_2\left(c_s^2 \frac{\partial \rho}{\partial x_1} - \frac{\partial p}{\partial x_1}\right) \tag{10–2.11}$$

$$\mathcal{L}_3 = \lambda_3\left(\frac{\partial v_2}{\partial x_1}\right) \tag{10–2.12}$$

$$\mathcal{L}_4 = \lambda_4\left(\frac{\partial v_3}{\partial x_1}\right) \tag{10–2.13}$$

$$\mathcal{L}_5 = \lambda_5\left(\frac{\partial p}{\partial x_1} + \rho\, c_s \frac{\partial v_1}{\partial x_1}\right). \tag{10–2.14}$$

The $\{\mathcal{L}_i\}$s may be interpreted as the amplitude variations of the characteristic waves crossing the boundary. Note that the viscous terms do not appear in equations (10–2.10) through (10–2.14), but they are in **d** (equation (10–2.6)).

We want to use equations (10–2.1) through (10–2.5) to advance the system on the boundaries. Most of the quantities in these equations can be estimated using the interior points and values at the previous timestep. The parallel terms may be found on the boundaries with the same approximations used in the interiors. The terms that require care involve the elements of **d**, which are functions of $\{\mathcal{L}_i\}$ and contain diffusion terms. The $\{\mathcal{L}_i\}$ may be calculated using one-sided derivatives.

For example, the waves propagating through the boundaries from the inside to the outside may be computed from one-sided discretizations, as we know the inside values. The waves propagating through the boundaries from the outside to the inside are more of a problem, and require additional input or a one-dimensional analysis. Specifically, Poinsot and Lele (1992) estimated these by neglecting the transverse and diffusive terms, and finding the local, inviscid $\{\mathcal{L}_i\}$. That is, they solved a set of equations here of the general form

$$\frac{\partial \rho}{\partial t} + d_1 = 0 \tag{10–2.15}$$

$$\frac{\partial p}{\partial t} + d_2 = 0 \tag{10–2.16}$$

$$\frac{\partial v_1}{\partial t} + d_3 = 0 \tag{10–2.17}$$

$$\frac{\partial v_2}{\partial t} + d_4 = 0 \tag{10–2.18}$$

$$\frac{\partial v_3}{\partial t} + d_5 = 0 \tag{10–2.19}$$

in addition to an equation for $\partial T/\partial t$ that involves the $\{\mathcal{L}_i\}$.

The characteristic approach described here has been shown to work well in multidimensional Navier-Stokes problems, and, with some extensions, can be used for flows that include species reactions and energy release. When problems arise, they can usually

be traced back to the approximation used for the $\{\mathcal{L}_i\}$ and the fact that there are no true two- or three-dimensional characteristics. For example, in multidimensional computations involving steady inflow, constant temperature walls, and an outflowing boundary layer, the one-dimensional approach described above worked well except for inflows that were not parallel (such as stagnation points) and for outflow in the viscous boundary layer. These problems could be fixed by writing equations (10–2.1) through (10–2.5) in two dimensions, again including all of the viscous terms.

Other problems arise for multidimensional reactive flows with substantial energy release (Ott 1999). In this case, using the standard outflow formulation produced instabilities that propagated into the computational domain. This was handled by extrapolating the energy density and mass density at the outflow boundary. The velocities were still updated from the basic equations at the boundaries, taking the characteristic waves into account. This problem is related to the fact that the original boundary-condition model was developed for a calorically perfect gas and works well for cold, fairly uniform flows. For more complex flows, the boundary conditions require modifications.

10–2.2. Problems with Acoustic Radiation in CFD

Properly resolving acoustic waves in computational fluid dynamics has now become quite important. Acoustic-flow interactions are a topic of major concern in aeroacoustics, where it is required to compute jet and flow noise and thus evaluate flow and structural modifications that reduce it. In thermoacoustic problems, the generation of very small acoustic impulses is used to change and control the macroscopic behavior of flow systems, often in the presence of combustion. These computations require high-order, preferably monotone, central-difference algorithms that can resolve the sound waves, acoustic spectra, and the fluid flow itself. In such problems, methods for treating boundary conditions must account for the details of sound waves themselves passing through boundaries of systems containing flows. Usually these solutions cannot tolerate even small spurious reflections, damping, or instabilities at boundaries.

A number of different types of boundary conditions have been used for aeroacoustics. One method is to use characteristic boundary conditions as described above. These work well for acoustic disturbances incident nearly normal on the boundary, but there are problems with grazing incidence at the boundary or when there is strong mean flow tangential to the boundary (Tam 1998). These problems may also be treated by using multidimensional forms of equations (10–2.1) through (10–2.5).

Another interesting approach involves the asymptotic solution of the governing equations (Bayliss and Turkel 1980, 1982; Hagstrom and Hariharan 1988; Tam and Webb 1993). This approach superimposes small-amplitude disturbances on a uniform mean flow, and then finds the linearized Euler equations. It was found that the results for the radiation and outflow boundary conditions are good as long as the sources are far enough from the boundary of the computational domain. When the sources are close to the boundary, the solutions are degraded. Yet another approach uses a spongelike absorbing layer, which is sometimes ten or twenty computational cells thick, much like the Rayleigh viscous zone discussed above. Another approach is a method originally developed for electromagnetics, called the perfectly matched layer (PML) Berenger (1994, 1996). The idea here is that an additional artificial damping term is added that damps incident disturbances. It has been

shown that it is possible to formulate an absorbing layer without reflection. A currently unsolved problem is that the PML equations with a mean flow are unstable.

In addition to the problems of inflow and outflow, the presence of acoustic waves may generate spurious solutions at the wall. For aeroacoustics problems, there are two types of spurious waves: propagating waves with short wavelengths and waves that are spatially damped. These spurious waves are particularly evident in higher-order finite-volume algorithms that generate short-wavelength numerical noise at boundaries. Therefore, when an acoustic wave reaches a wall, there are both the expected physical reflections and these spurious numerical waves. Spatially damped waves are also generated, but they decay as they propagate away from the wall, and effectively form a numerical boundary layer on the surface. The problems caused by these spurious waves have been addressed for specific problems of boundary-layer interactions and thermoacoustic waves by Weber et al. (1995) and Farouk et al. (2000), respectively.

10–3. Interfaces and Discontinuities

An interface inside a computational domain refers to a boundary that separates two regions representing different materials or different states of matter. Though it is common to speak of discontinuous physical properties at an interface, there are no true discontinuities. Physical interfaces have finite, though relatively small widths. The very steep gradients associated with the interface create a computational problem because of the range of spatial scales involved.

We distinguish two types of interfaces. A *passive interface* is a moving surface that models the physical "discontinuity" as it moves through the fluid. Its behavior is solely determined by its location and the motion of the fluids on both sides. An *active interface* usually has a finite thickness and displays characteristics unlike either of the bordering materials. An active interface not only advects with the flow, but it can also interact with the fluids on either side or advance through them. A flame or solidification front is an example of an active interface.

Whether an interface is active or passive is sometimes ambiguous and depends on how accurately the transition from one state to the other is represented. An example of a passive interface could be the border between an inert liquid and the air flowing over it. This same interface could be active in a molecular dynamics calculation of the interaction between the evaporating molecules of the liquid and the air molecules impinging upon it. A passive interface, in the limit of slow burning for example, could represent a flame front as a surface of discontinuity separating reacted and unreacted material. If the chemical reactions in the flame cause it to advance through the reactants, or if the finite thickness and gradients in the flame are resolved, this same interface becomes active.

The art and science of resolving and computing interfaces has developed significantly in the past ten years. There have been major advances in both the mathematical formalism and in the direct application to specific science and engineering problems. Many types of computations are only possible by combining accurate interface methods with solutions of the appropriate Euler or Navier-Stokes equations for the fluids bounding the interface.

Resolving the physical properties of an interface in a simulation is often called *interface capturing*. The properties of the interface are described by models of the physical processes that create and maintain the interface, and the results may be coupled to the flow, either

actively or passively. Like shock capturing discussed in Section 10–3.1, the microscopic details of the interface may not be fully resolved, but the integral properties of the interface must still be accurately represented. Usually, when we resolve an interface, it is an essential active aspect of the calculation. For example, a flame front may be modeled as an active surface resolved by a fine, adapting mesh. Descriptions of active interfaces involve the numerical issues of resolution, accuracy, and coupling different physical models, which are the subjects of Chapters 6, 8 and 9, and 11, respectively. Passive interfaces, however, can also be captured.

It is usual to describe an interface procedure as *interface tracking* when it models the motion of a sharp, unresolved interface. Special algorithms are used to represent and determine the spatial location of the interface as it cuts through contiguous computational cells. Again, interface tracking can be applied to either active or passive interfaces. Even when the contributions from diffusive physical processes, chemical reactions, or radiation cannot be computed self-consistently, an advancing flame front may still be tracked phenomenologically as an active surface that propagates through the unreacted gas using a predetermined set of rules. Between the two extremes of fully resolving an interface and tracking a discontinuity with a phenomenological set of rules, there are different levels of detail in models that can make distinctions between interface capturing and interface tracking somewhat ambiguous. In fact, some computations are best performed with a combination of these approaches.

Interface tracking, as a separate field from adaptive gridding, has now become better formulated. There are several classes of methods used to track interfaces numerically. These include moving-grid, surface-tracking, and volume-tracking methods, each of which is discussed below. Moving grids were discussed in some detail in Chapter 6. Good general references include those by Hyman (1984), Hirt and Nichols (1981), Sethian (1996), and Shyy et al. (1996).

10–3.1. Resolving Active Interfaces

The only way to capture an active interface is to resolve the controlling physical processes at some useful level of fidelity. Chapter 6 described and gave examples of many methods for achieving enough local resolution to resolve an interface. The simplest example is the Eulerian sliding rezone, shown in Figure 6.9, which has been used for locally increasing the resolution in shock and detonation computations. Figure 6.7 showed a number of approaches for gridding complex geometries. These techniques, which include block-structured, overset, unstructured triangle-based, and global Cartesian grids, can all be used to provide resolution at an interface. AMR algorithms, as shown generally in Figure 6.13 and in the example of the deflagration-to-detonation computation shown in Figure 6.14, have been used to resolve flames, shocks, detonations, and contact surfaces, all in the context of complex and highly dynamic systems. AMR applied to reacting flows in supersonic nozzles, with multiple, time-varying shock fronts and contact surfaces, has been carried out using finite-element methods on an adaptive grid. An example is shown in Figure 6.17.

An example of a resolved complex flame interface is shown in Figure 10.6, which is taken from a computation of a flickering methane-air diffusion flame (Kaplan, Patnaik, and Kailasanath 1998). The methane exits a nozzle into an air background. In addition

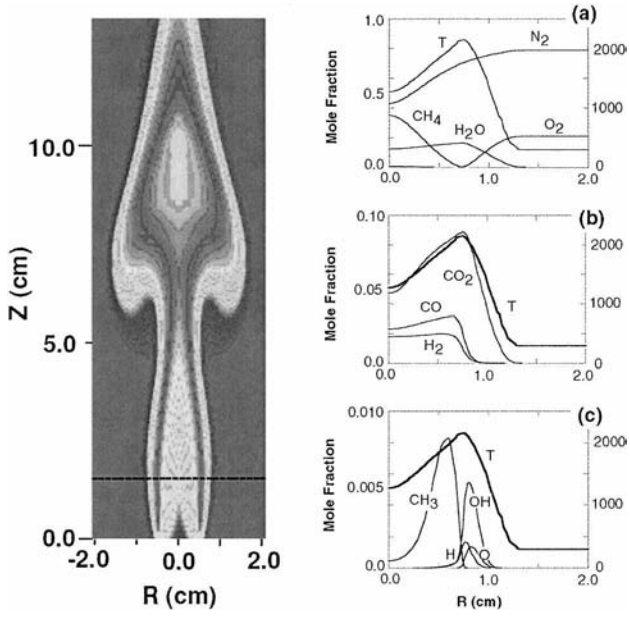

Figure 10.6. Simulation of an axisymmetric, flickering, methane-air diffusion flame taken from work of Kaplan et al. (1998). The left figure is a temperature map of the flame. Computed profiles of the mole fraction of the major species, at the horizontal location marked by the dashed line (approximately 2 cm), are shown on the right side in (a), (b), and (c). Note the difference in the mole-fraction magnitudes along the y-axes. Not shown are the much smaller concentrations of CH_2O, HO_2, H_2O_2, CH_3O, and HCO. T = temperature profiles.

to basic compressible fluid dynamics, these computations included models of thermal conduction, molecular and viscous diffusion, and a fairly complex, multispecies reaction mechanism that described the chemical reactions and heat release. A fine grid was used around the flame. The temperature map on the left, taken from one timestep in the calculation, shows the bulge, starting at around $Z = 6$ cm, typical of a flickering flame. Profiles of the concentration of various chemical species at $Z = 2$ cm are shown on the left. The example of an axisymmetric flame is used again in Chapter 13 to illustrate the effects of radiation transport and to discuss features of coupling many physical processes in a chemically reacting flow.

Shock Capturing

Consider a grid-based calculation of a shock propagating through a gas. The physical shock profile is only a few mean free paths thick. For continuum flows, this is orders of magnitude too small to be resolved in a macroscopic fluid calculation. Nevertheless, most algorithms now used to simulate shocks are *shock-capturing algorithms* (see Section 9–2), because the physical conservation conditions leading to the correct shock strength and speed parameters are properly represented and solved. Numerical diffusion, flux limiters, or artificial viscosity play the role on the macroscopic scale that molecular viscosity plays on the microscopic scale. A few cells are used to represent the shock profile. The shock is "captured" in the sense that the shock discontinuity lies near the middle of the steep

numerical gradient. The shock is not well resolved in the detailed sense as accurately as the fully resolved flame front described above, and yet the conservation form of the equations ensures that the correct physical jump conditions across the shock are satisfied. Thus shock capturing also ensures that shocks and contact surfaces move through or with the fluid at the correct local speed.

Computation of shocks and shock interactions were illustrated in Figure 6.8, which showed the result of passing a planar shock through the Euler equations. Figure 6.14 shows computations of shocks in turbulent reacting flows, and Figure 6.17 shows supersonic flows through a nozzle. These calculations used the shock-capturing algorithms described in Sections 8.4 and 9.2, implemented on adaptively refined stationary or moving grids. If the resolution at the shock front is increased enough, or the conditions of the flow are such that the shock is physically broadened, the same shock-capturing method shows more of the structure inside of this interface.

Shocks can also be advanced by shock-tracking algorithms where the discontinuity is treated as a moving geometric discontinuity in space. These methods localize the shock more precisely than shock-capturing methods but do not lend themselves easily to problems with more complex physics and geometry. See, for example, Moretti and Abbett (1966) for different approaches and applications. The speed and jump conditions across the moving discontinuity are computed analytically from the fluid states behind and in front of the shock by solving the appropriate Riemann problem and considering the flow characteristics. The advantage is a very precise definition of the shock location. The disadvantage is rapidly increasing programming complexity as the topology of the shock surface changes discontinuously when new shocks form or multiple reflections occur. Each shock reflection and transmission is not too difficult to compute, but it is daunting to track all of the new shock and contact surfaces that are forming and colliding at arbitrary angles.

Flame Capturing

The basic concept of shock capturing is that the numerical solution naturally produces, but not necessarily resolves, the correct properties of the interface. The numerical methods do this by using the numerical dissipation that occurs naturally in the solution of the Navier-Stokes or Euler equations to replace the actual physical dissipation occurring at an unresolved scale. This general idea has also been extended to flame capturing in several different ways. Unlike shock-capturing methods, flame-capturing methods must introduce an additional equation to describe the movement of the flame front. The key question in this case is how to build an equation that will represent the correct properties of the flame on the chosen grid, without resolving the flame thickness exactly. The model must still produce flame solutions that are independent of the grid resolution and orientation. Another similarity between flame-capturing and shock-capturing methods is that neither defines the exact location of the interface, but represents the captured interface as a quantity that changes continuously but steeply over several cells.

An early example of this approach is the *gradient method* (Laskey, Oran, and Boris 1987). This method solves an equation of the form

$$\frac{\partial n_p}{\partial t} + \nabla \cdot (\mathbf{v} n_p) = \dot{w}, \tag{10–3.1}$$

where n_p is the number density of combustion products, and $\dot{w}$ represents the production

of n_p. The reaction fronts are identified by the region where the gradient ∇n_p is large. The integral of the gradient from the front to the back of the extended interface – that is, the change in n_p across the flame – is known because the properties of a laminar flame are assumed to be known. Choosing the local energy-release rate proportional to the local gradient guarantees the correct integrated energy release along the convoluted front. The amount of new product formed is

$$\Delta n_p = |\nabla n_p| v_f \Delta t, \tag{10–3.2}$$

where Δn_p is the change in n_p, v_f is the known local flame speed normal to the interface, and Δt is the timestep. The quantity $v_f \Delta t$ is the distance the front moves normal to itself during the timestep. The direction of the normal to the reaction front is the same as the direction of the gradient. The speed of the front is the local flame speed. The amount of product formed per unit volume in the time interval Δt is

$$\Delta w \equiv \Delta n_p = v_f \Delta t |\nabla n_p|. \tag{10–3.3}$$

The left side of equation (10–3.1) can be solved by any continuity-equation solver, preferably one consistent with the algorithms used to solve the Navier-Stokes equations. This flame-transport term is added to the convection by process-splitting methods (see Chapter 11). The condition for conservation of reactants and products determines the amount of energy released in each cell. The issue remaining is how to compute $|\nabla n_p|$, which involves taking a numerical derivative. This choice is generally problem dependent, and might be selected to be of the same order as the method used to solve the continuity equation.

An extension and expansion of these concepts has been developed for three-dimensional flame computations (Khokhlov, Oran, and Wheeler 1996). This approach also builds in the known flame speed and energy release in the form of an additional equation that describes the motion of the propagating laminar flame. In addition to the Euler equations, an additional reaction-diffusion equation is solved for the mole fraction of reactant, f,

$$\frac{\partial f}{\partial t} + \mathbf{v} \cdot \nabla f = K \nabla^2 f + R, \tag{10–3.4}$$

where K and R are a diffusion coefficient and reaction rate, respectively, and $f = 0$ represents pure fuel and $f = 1$ represents pure products. The quantities K and R are expressed in terms of v_f, the known energy release across the flame front, and various numerical parameters such as the computational cell size, all of which ensure the proper behavior of a propagating flame using a minimum number of cells. Solving equation (10–3.4) requires solving a continuity equation, a reaction term, and a diffusion term. These processes are solved separately and combined by process-splitting techniques (see Chapter 11). Extensive tests were performed to show the grid-independence and the invariance of the flame propagation to the orientation of the grid. The captured flame is spread over about three computational cells. Figure 10.7 is taken from a three-dimensional computation that used this method to propagate a flame. This approach is an example of the ambiguity that exists in classifying models as flame tracking or flame capturing. Depending on your point of view, this model could be called either.

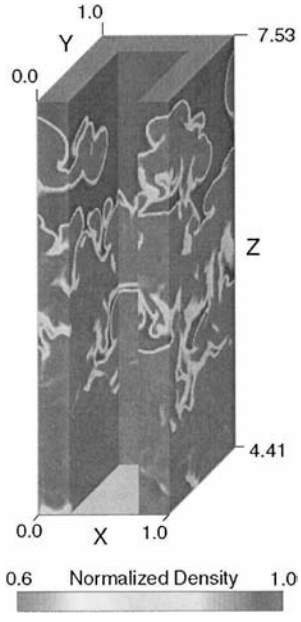

Figure 10.7. Normalized density taken from three-dimensional computations of the structure of a turbulent flame. The density is normalized by the initial density of the fuel. The flame is propagating from the bottom to the top of the domain. These computations were performed to derive a scaling law for the terminal velocity of the turbulent flame in a situation in which diffusion effects are small compared to the fluid-dynamics effects, and turbulence is driven by the Rayleigh-Taylor instability (Khokhlov et al. 1996).

There are several useful features of flame-capturing methods. They ensure that the right amount of reaction takes place in the vicinity of the front, as defined by the macroscopic grid, and they can be calibrated to any combination of reacting species. Flame-capturing methods treat merging interfaces with relatively little difficulty. Because no additional variables are needed, computer memory requirements are modest. Flame-capturing methods are inexpensive and may be optimized easily on any type of scalar, vector, or parallel computer. Their application in one, two, or three dimensions is straightforward. These methods work well for the types of problems for which they were designed, for example, reaction-diffusion fronts propagating in complicated flow fields.

When using a flame-capturing method, you do not know the exact location of the interface. Thus, such methods are not suitable for simulations that must rigorously separate two materials and track the curvature of the interface on scales comparable to or smaller than the grid spacing. An example of this latter type of problem is the dynamics of a droplet in which it is important to know the surface curvature very accurately to compute the dynamic effects of surface tension. Flame-capturing methods, however, can locate the interface accurately enough to extract information about flame behavior that would be useful for describing a droplet surface and the spread of vaporized material into the background gas. Methods for more accurately tracking an interface are summarized in Section 10–3.2.

Detonation Capturing

For many important practical problems involving deflagrations and detonations, computations that would resolve the reaction zone and structure of a detonation are prohibitively expensive. The resulting question is how to solve the reactive Euler equations, with the correct energy-release model, and compute a detonation wave that propagates at the correct velocity. A computational physics problem arises because it is possible for too much

energy to be released in the computational cell in a single timestep, thus causing the computed detonation to propagate at an unphysically high speed.

We treat this as a coupling problem whose solution is an issue in properly coupling compressible fluid dynamics and an energy-release model. Chapter 11 (Section 11–3.1) discusses the origin of this problem and suggests simple, physically motivated ways of limiting the energy release at the detonation front.

10–3.2. Interface Tracking with Moving Grids

Moving-grid methods adjust the grid so that a moving interface is always located on well-defined cell boundaries. The issue here is to control the fluxes going through cell interfaces in a way that models the correct interface physics. In some cases, the interface is a rigid, passive, nonreacting, reflecting wall. In others, it may be an ablating or advancing surface. In others, it is a contact surface that can deform with the flow.

Gridding methods appropriate for treating these types of problems have been discussed in Sections 6–3 and 6–4 and are an area of intense development. Different approaches have their own sets of advantages and disadvantages. The common advantages are a potentially good representation of the interface. A common disadvantage is potential reflections and unphysical waves in the solution due to the moving grid. Unstructured, triangle-based grids seem to be a natural approach, although they are complicated to implement. Quadrilateral moving grids, with interfaces cutting through the grid, may at some point rival unstructured grids and body-fitted grids, but they need substantial development. Another approach is a combined Eulerian-Lagrangian approach that could be achieved by continually defining a new grid and interpolating the physical variable onto that grid. This interpolation introduces numerical diffusion, which could be a problem. Producing adaptive grids containing moving bodies continues to be an area of intense development.

10–3.3. Surface-Tracking Methods

Surface-tracking methods to locate and track an interface include level sets, boundary elements, and contour dynamics, as well as a number of hybrid approaches that combine various elements of these methods. Again, this broad category of methods has a number of features, procedures, and applications that overlap with each other and with the other methods described in this section.

The basic idea of the original surface-tracking methods is to represent an interface as a connected series of curves interpolated through points defining the interface. At each timestep, the points are advected with the flow, and the sequence in which they are connected is saved. The points can also move to simulate processes other than convection, such as chemical reactions, phase changes, or ablation. In simple surface-tracking methods for two dimensions, the points are saved as a sequence of heights above a given reference line, as shown in Figure 10.8a. Two curves are shown bounding the top and the bottom of a region of fluid. This approach fails if the curve becomes multivalued or does not extend all the way across the region. Such failure may be avoided if the points are parametrically represented, as that shown in Figure 10.8b. The formulation is more complex, but it can represent fine detail in the interface if enough points are used. An important feature is that

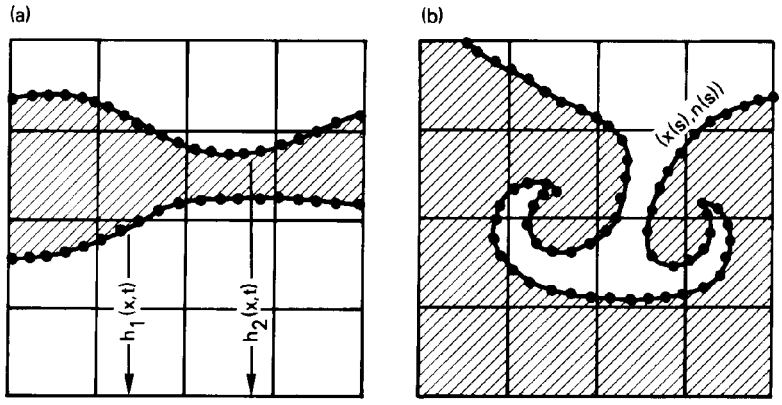

Figure 10.8. The interface may be determined by (a) a sequence of heights above a reference line, or (b) a series of points parameterized by a distance along the interface (Hyman 1984).

surface-tracking methods can resolve features of the interface that are smaller than the cells of the Eulerian grid on which the curves are overlaid. There is naturally a price paid for storing this additional information, and the timestep is limited by how much change can be tolerated at each timestep on the smallest resolved space scale.

There are a couple of difficulties in using surface-tracking methods. First, it is difficult to handle merging interfaces or to join a part of an interface to itself. These situations require detecting the crossing, reordering the interface points (which could involve significant computational bookkeeping), and possibly launching and tracking additional interfaces. Second, the points can accumulate along one portion of the interface leaving other portions without enough resolution. Because interface areas typically increase continually in complex flows, it is necessary to add points along the interface automatically. Conversely, points should be deleted where there are too many. For a particular application, there is always a question of the best way to interpolate new points and the best way to represent and manipulate contours with changing length scales. Progress in this area is coupled to improved methods for dealing with changing topology from simply connected to multiply connected regions, and to new algorithms for merging fronts, eliminating weakened fronts, and generating interfaces at new fronts.

Boundary-element methods (*BEMs*) are based on an integral equation formulation of boundary-value problems. BEMs require solving a descretized set of equations involving only the domain boundary, either a curve or a surface, and not for the continuum solution on the entire interior of the domain. Thus they appear to lead to a reduced-order set of equations. The evolution equation for the curves that separate the different regions is written as a line integral along the curves. The prescribed boundary conditions are used to connect the unknown boundary values to the known values. Then, to track these contours, points or nodes are defined on the curves, and the motions of these points are integrated. If the flow becomes too complex and the contours are too distorted, the points along the curve must be redistributed. In one formulation of BEMs, the boundary equations are formulated by using a related field equation. This connects the method to contour dynamics (see Section 9–7), which is a Lagrangian approach for tracking vorticity in incompressible flows. To determine the fluid velocity, contour dynamics solves a line integral along a surface that separates different constant-vorticity regions.

BEMs are used for problems with complicated internal boundaries and unbounded regions where the physics and fluid dynamics between the boundaries are particularly simple. Volumetric distributions of vorticity, for example, usually invalidate the use of BEMs. A major advantage is a significant reduction in the computing required. Again, a key issue is how to redistribute the nodes for a particular problem. A major problem area is that the solutions may be prone to errors that are directly related to difficulties in solving integral equations. Good summaries of BEMs may be found in the texts by Brebbia and Domingues (1992) and Kythe (1995).

The level-set approach (Osher and Sethian 1988) has features similar to those described for resolved and captured interfaces in the use of a continuum partial differential equation on the entire volume to propagate the interface. Level-set methods extend interface-capturing approaches by rigorously identifying the interface as a particular contour with constant value ("a level set") of a continuous function, ϕ, which moves with the fluid and can propagate normal to its surface. Then, for example, if $0 \leq \phi \leq 1$, the interface may be defined as the contour surface $\phi = 0.5$. The issue in each particular application is to determine the appropriate ϕ and the equation that propagates it. Level-set methods typically propagate ϕ on an Eulerian grid. They present a formalism for relating and describing flame-capturing techniques.

Level sets have been applied to broad classes of problems, including descriptions of free surfaces, bubbles and drops, Rayleigh-Taylor instabilities, and crystal growth. A number of these are summarized by Sethian (1996). One advantage of this method is that once ϕ is selected, there are convenient formulas for finding the normal to and the curvature of the interface. Another advantage of this method is that no special procedures are required to model topological changes in the front: the value of ϕ is just updated at each timestep, and the location of the front at the new time is the contour $\phi = 0.5$. A problem can arise with conservation. Conserving ϕ does not imply that the volume of each type of material is conserved. Some form of mass renormalization is one way to deal with this. Another problem arises with very convoluted contours. Incompressible, turbulent, small-scale convection, for example, may spread ϕ over such a large volume in which the value of ϕ is never greater than 0.5. Nonetheless, half of the volume is still filled with one fluid, and half with the other fluid, and they are coarsely intermixed.

The G-function is another approach to propagating a laminar flame. Williams (1985) derived an equation for a scalar function $G(\mathbf{x}, t)$ whose level surfaces, $G(\mathbf{x}, t) = G_0$, represent the flame surface,

$$G(\mathbf{x}, t) - G_0 = x + F(y, z, t), \tag{10–3.5}$$

where $\mathbf{x} = (x, y, z)$ are spatial coordinates, $F(y, z, t)$ is the displacement of the flame surface from its mean position towards the unburned gas. Then $G(\mathbf{x}, t) - G_0$ is the x-distance between the flame and the level surface and defines a localized spatial coordinate at the flame surface. A *scalar-field equation*, or *G-equation*, is used to advance G,

$$\rho\left(\frac{\partial G}{\partial t} + \mathbf{v} \cdot \nabla G\right) = (\rho S_L)|\nabla G|, \tag{10–3.6}$$

where ρS_L is the burning velocity through a laminar, plane, steady flame. As G can be related to a mixture fraction, the method takes on a form similar to several of those

described above. The G-equation approach was solved numerically by Kerstein, Ashurst, and Williams (1988) in simulations of propagating flamelets in a turbulent flame. A good summary of the method is found in Bray and Peters (1994).

10–3.4. Volume-Tracking Methods

Volume-tracking methods are used extensively because they are reasonably accurate and relatively simple to implement. They include the well-known methods referred to as MAC, PIC, SLIC, and VOF. Unlike surface-tracking methods that store a representation of the interface at each timestep, volume-tracking methods reconstruct the interface whenever it is needed. The reconstruction is done cell by cell and is based on the location of a marker within the cell or on some other volumetric progress variable. Whereas the surface-tracking methods represent the interface by a continuous curve, the interface generated by volume tracking consists of a set of disconnected line segments (two dimensions) and surfaces (three dimensions) in the cells containing parts of the interface. There have been many variations on this approach presented in the literature for a wide range of applications. Level sets, G-equations, flame capturing, and the gradient method may all be considered specific evolutions of this class of approaches.

Rider and Kothe (1998) give a good review of volume tracking that has a historical perspective. They point out that there are two important features that differentiate the various standard volume-tracking methods: how they reconstruct the interface, and how they advect the volume. The reconstruction algorithms are either piecewise linear or piecewise constant; the volume advection algorithms are either operator split or multidimensional. Each method uses some combination of these and a different method to reconstruct the interface geometry.

The earliest volume-tracking methods used marker particles whose number density in each cell indicated the density of the material. This method, first proposed by Harlow (1955), was called the *particle-in-cell (PIC) method*. In the *marker-and-cell (MAC) method* (Harlow and Welch 1965; Welch et al. 1966), the particles are tracers, that is, marker particles with no mass. Later the marker particles were changed into marker functions creating, for example, the *volume-of-fluid (VOF) method* (Hirt and Nichols 1981).

Consider some of the features of the PIC method implemented by Amsden (1966) using an Eulerian grid in which velocity, internal energy, and total cell mass are defined at cell centers. The different fluids are represented by Lagrangian mass points, the marker particles, that move through the Eulerian grid. The marker particles have a constant mass, a specific internal energy, and a recorded location in the grid that is updated using the local cell velocity. The particle mass, momentum, and specific internal energy are transported from one cell to its neighbor when the marker particle crosses the cell boundary. Cells containing marker particles of two fluids contain an interface. Because the interface can be reconstructed locally at any time, the problems associated with interacting interfaces and large fluid distortions are reduced and the method generalizes to any number of fluids.

Many volume-tracking methods now use the fraction of a cell volume occupied by one of the materials to reconstruct the interface. If this fraction is zero for a given cell, there is no interface present. Conversely, if the fraction is one, the cell is completely occupied by the material and again there is no interface present. An interface is present only if the fractional marker volume is between zero and one.

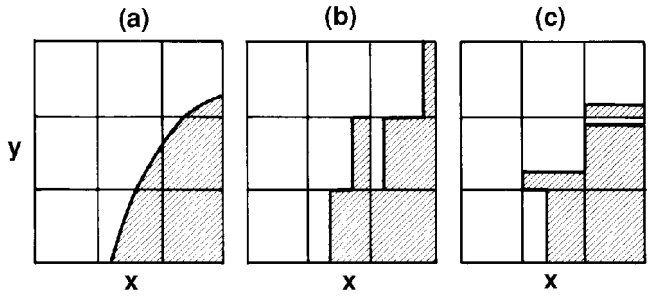

Figure 10.9. (a) The actual curved fluid interface. (b) The SLIC representation on the x-pass. (c) The SLIC representation on the y-pass. (Noh and Woodward 1976.)

In the *simple line interface calculation* (*SLIC*) algorithms (Noh and Woodward 1976), each cell is partitioned by a horizontal or vertical line such that the resulting partial volume equals the fractional marker volume. The orientation of the line is chosen so that, if possible, similar types of fluid in neighboring cells are adjacent. The orientation is chosen to give lines through cells normal to the direction of flow. This minimizes numerical diffusion by having all of one type of fluid move across a cell boundary into a region of like fluid, before another type of fluid can enter a cell. The method assumes timestep splitting, so that extensions to two or three dimensions are straightforward. For the two-dimensional case, the interface is constructed cell by cell before advection in the x-direction. After the x-integration, the interface is reconstructed and the y-integration is done. Line segments normal to the direction of flow result in different representations of the interface for the x- and y-sweeps. To avoid such a directional bias, the order of x- and y-integrations is changed every timestep. Samples of SLIC interface approximations are shown in Figures 10.9 and 10.10. In SLIC algorithms, the interface reconstruction is piecewise constant, and the interfaces in each cell are assumed to be lines or planes aligned with one of the mesh coordinates.

Chorin (1980) improved the resolution of the original SLIC algorithm by adding a corner interface structure to the straight horizontal and vertical lines and using the same fractional cell volume. An example of this SLIC interface is shown in Figure 10.11b. Chorin used the vortex-dynamics method (see Section 9–7) to calculate the vorticity. The velocity derived from the vorticity was used to advect the interface. Advection is done in the same time-split manner as in the original SLIC algorithm.

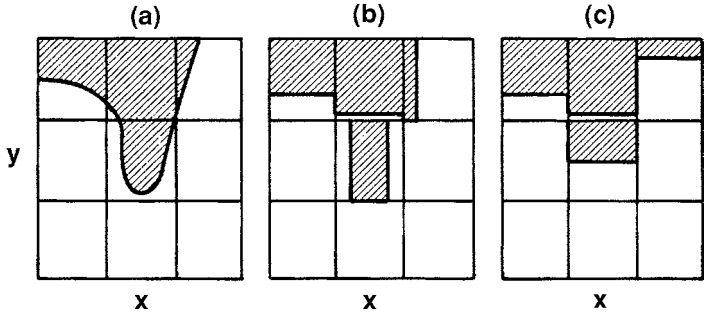

Figure 10.10. (a) The actual interface. (b) The SLIC representation on the x-pass. (c) The SLIC representation on the y-pass. (Noh and Woodward 1976.)

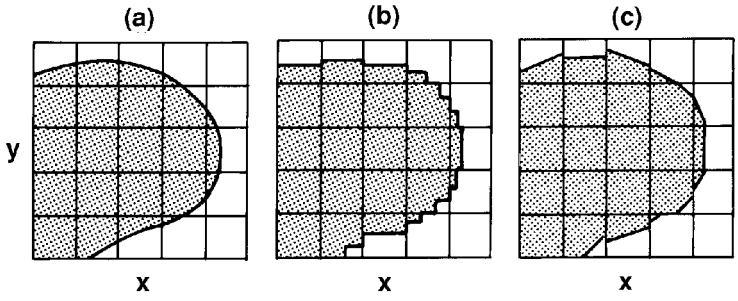

Figure 10.11. (a) The actual interface. (b) Reconstructed SLIC interfaces for convection (Chorin 1980). (c) Reconstructed interface for boundary conditions using VOF. (Barr and Ashurst 1984.)

Barr and Ashurst (1984) give a detailed discussion of difficulties with using SLIC algorithms. In brief, curved surfaces may be flattened or even indented. This distortion depends on the Courant number of the flow and the interface geometry. In some cases, this distortion appears as the interface first moves across a cell and then does not increase. In other cases, the distortion may continue to grow. Including a model for propagating chemical-reaction fronts apparently decreases this distortion when the propagating reaction front smooths short-wavelength wrinkles.

VOF (Hirt and Nichols 1981) also represents the interface within a cell by a straight line, but the line may have any slope. A numerical estimate of the x-direction and y-direction derivatives of the volume fraction (the progress variable) are obtained for a given cell. If the magnitude of the x-direction derivative is smaller than the magnitude of the y-direction derivative, the interface is closer to horizontal and its slope is the value of the x-direction derivative. A more nearly vertical interface has a slope equal to the value of the y-direction derivative. If the x-direction derivative gives the slope of the interface, then the sign of the y-direction derivative determines whether the fluid is above (positive) or below (negative) the interface. For a more vertical line, the sign of the x-direction derivative determines whether the location of the marker fluid is to the right (positive) or the left (negative) of the interface. Given the slope of the interface and the side of the interface on which the fluid is located, the position of the interface within the cell is set. These calculations are done for every cell with an occupied volume between zero and one. Hirt and Nichols used values of the fractional volume of fluid averaged over several cells to calculate the derivatives.

Other VOF algorithms use simple central differences. VOF methods depend on the ability to advect the volume fraction through the grid accurately without smearing from numerical diffusion. Hirt and Nichols have described a "donor-acceptor" method that ensures that only the appropriate constituent fluid moves to or from a cell containing an interface. This helps to avoid the cell averaging that results in numerical diffusion.

Barr and Ashurst (1984) use both VOF and the Chorin version of SLIC in a method called SLIC-VOF for flame tracking. The SLIC algorithm is used to define and advect interfaces and VOF is used to define the normal direction and to give a smoother interface for the flame propagation phase. Figure 10.11 shows how SLIC and VOF approximate the same curved interface. For a two-dimensional flame-propagation problem, SLIC-VOF performs the x-integration of the SLIC interface, the y-integration of the SLIC interface,

the x-direction flame propagation of the VOF interface, and then the y-direction flame propagation of the VOF interface. The interfaces are reconstructed following each advection or propagation calculation. At each full timestep, the order of the x- and y-sweeps are interchanged, but convection always precedes burning. As in the Chorin implementation, flow velocities are calculated from a vorticity field. The flame propagation speed in the x- and y-directions is the sum of the flow velocity and the flame speed directed normal to the interface defined by VOF.

Rider and Kothe (1998) pointed out that expressions for the volume fluxes used in VOF can be formulated algebraically, without needing an exactly reconstructed interface position. The volume fluxes can be expressed as a weighted sum of upwind and downwind contributions, depending on the orientation of the interface relative to the local flow direction. If the reconstructed interface is parallel to the flow, an upwind flux is used. Otherwise, a portion of downwind flux is added to counteract numerical diffusion brought about by the piecewise-constant upwinding. From this observation, Kothe and Rider noted that VOF falls into the general family of flux-corrected transport methods (see Chapter 8), and that this is an underlying theme behind the design of VOF methods. This idea was developed by Rudman (1997) who developed a VOF-FCT method and compared a number of the VOF methods in a series of difficult problems, including shears.

10–3.5. Some Concluding Remarks

The entire discipline of interface capturing and tracking is a complicated subject. As pointed out above, distinctions between the different continuum approaches can seem somewhat artificial and depend on the point of view. Using a single continuum tracer function defined over the whole domain, such as the density of water, underlies many of these methods. Then, for example, where the value of this function is close to zero, the material must be water. Where the value is close to one, the material must be air. The various methods differ in how they determine the location of the interface. In interface-capturing methods, it is permissible for the interface to be spread out over a couple of cells. This is viewed as the discrete, finite-resolution, numerical analogue to the finite molecular-scale transition that really exists. It is permissible to have air and water occupying the same volume because the real transition is continuous at the appropriate small scale.

Using the same tracer function, however, interface-tracking methods such as SLIC and VOF reconstruct a rigorously discontinuous interface. Special algorithms try to ensure that all of one fluid leaves a volume before any of the second fluid enters. As Figures 10.9 and 10.10 show, however, the finite grid resolution appears as a different kind of ambiguity, a convoluted interface spreads discontinuously over a couple of cells. Level-set methods use an intermediate interpretation; the tracer function can be smooth and continuous while the level set defining the interface is mathematically sharp. This interpretation still runs into the same roadblocks for mixed media and droplets at the grid scale. Their manifestation is just a little different.

Unlike interface-tracking and interface-capturing approaches based on a volumetric tracer function, the spectrum of Lagrangian interface-tracking and related marker-particle methods can in principle resolve details of the interface that are smaller than the mesh size. Although the velocity fields may be determined on a grid, the interface structure

can be more resolved if enough tracer particles are used. Such tracer particles may accumulate in portions of the grid or on portions of the interface, thus leaving other portions poorly resolved. As described in Chapter 9 for CFD models based on particles, unacceptable statistical fluctuations arise when there are not enough Lagrangian degrees of freedom.

10–4. Matrix Algebra for Finite Differences

In this section, we outline techniques for solving the coupled algebraic and matrix equations that arise from the elliptic and parabolic equations describing fluid dynamics and heat transport. Generally the matrices are *sparse*, meaning that most of the off-diagonal elements are zero. Because they are sparse, using the most general matrix-inversion methods is usually not the best approach, and it is almost never required or recommended for solving these sparse matrices. The most general matrix-inversion algorithms scale as N^3, where N is the dimension of the vector unknowns. In the case of reactive flows, N is usually N_c, the number of computational cells, or it might be $N_c \times N_{sp}$, where N_{sp} is the number of reacting species. Typically, N will be in the range 10^5 to 10^9, which gives a *very* large matrix. Solving such a matrix *might* be done once, but it is a particularly odious task in a time-dependent problem, where the matrix equation would have to be solved once at every timestep. We can take advantage of the fact that the matrices are very sparse to use algorithms that scale at least as N^2. For certain problems, it is also possible to invert the linear matrices with algorithms that scale as $N \ln N$ or even N.

It is not our intent to give the details of the techniques. There are many good references, written at many levels, on numerical methods for solving matrix equations, in particular sparse matrices. These include the books by Nakamura (1991) and Ralston (1978). Good general references that may not appear in the numerical matrix and linear-algebra literature include Hockney (1970), Potter (1973), Roache (1982), and Tannehill, Anderson, and Pletcher (1997). A good introductory reference is Chapter 7 in Morton and Mayers (1994). The use of these algorithms for high-performance computing is discussed in the book by Dongarra et al. (1998).

10–4.1. Statement of the Problem

In many discretized representations of the reactive-flow equations, the partial-differential equations can be reduced to finite, linear, algebraic matrix equations. In explicit formulations, the equations usually become relations between diagonal matrices. In implicit formulations, the equations involve sparse matrices that must be inverted.

In the general case, simultaneous linear algebraic equations relate a vector of unknown quantities, $\mathbf{u}$, to a vector of known source quantities, $\mathbf{s}$, through a matrix $\mathbf{M}$,

$$\mathbf{M} \cdot \mathbf{u} = \mathbf{s}. \tag{10–4.1}$$

Assuming that there are N equations and N unknowns, $\mathbf{M}$ is a square matrix. Equation (10–4.1) has the formal solution

$$\mathbf{u} = \mathbf{M}^{-1} \cdot \mathbf{s}, \tag{10–4.2}$$

where the matrix $\mathbf{M}^{-1}$ is the $N \times N$ inverse of $\mathbf{M}$.

The one-dimensional diffusion equation (see Section 4–3) provides a good example,

$$\frac{d\rho}{dt} = D\frac{d\rho^2}{dx^2} + s. \tag{10–4.3}$$

With an evenly spaced grid having cell size Δ and a constant diffusion coefficient D, equation (10–4.3) (see equation (4–3.8)) can be differenced as

$$\frac{\rho_j^n - \rho_j^{n-1}}{\Delta t} = \frac{D\theta}{(\Delta)^2}[\rho_{j+1} - 2\rho_j + \rho_{j-1}]^n + \frac{D(1-\theta)}{(\Delta)^2}[\rho_{j+1} - 2\rho_j + \rho_{j-1}]^{n-1} + s_j^{n-1}. \tag{10–4.4}$$

Here j indexes each of the N computational cell values and the quantities in square brackets are evaluated either at the new time, n, or the old time, $n-1$. Equation (10–4.4) for the new values $\{\rho_j^n\}$ can be written symbolically as

$$\mathbf{M}^n \cdot \boldsymbol{\rho}^n = \mathbf{M}^{n-1} \cdot \boldsymbol{\rho}^{n-1} + \mathbf{s}^{n-1}, \tag{10–4.5}$$

where $\mathbf{M}^n$ and $\mathbf{M}^{n-1}$ are tridiagonal matrices. The generic form of a tridiagonal matrix is shown in Figure 10.12. Only the elements on the diagonal and adjacent to it are nonzero. In a periodic one-dimensional problem of the form of equation (10–4.4), the corner matrix elements X and X' are also nonzero, as indicated in Figure 10.12.

The one-dimensional Poisson equation,

$$\frac{d^2\phi}{dx^2} = -s(x), \tag{10–4.6}$$

also results in a tridiagonal matrix. In fact, when using algorithms coupling only nearest computational cells, any one-dimensional equation that can be put in the form

$$f\frac{d^2\rho}{dx^2} + g\frac{d\rho}{dx} + h\rho = s, \tag{10–4.7}$$

where $\rho(x)$ is the unknown and $f(x)$, $g(x)$, $h(x)$, and $s(x)$ are known functions of x, gives a tridiagonal matrix equation.

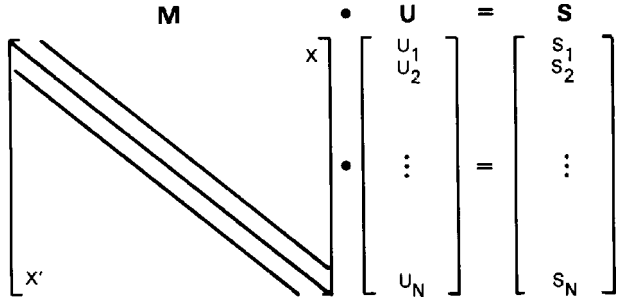

Figure 10.12. Generic form of a tridiagonal matrix. The slanted solid lines indicate nonzero coefficients, and the blank areas are assumed to be zero. The elements X and X' are zero unless the problem is periodic.

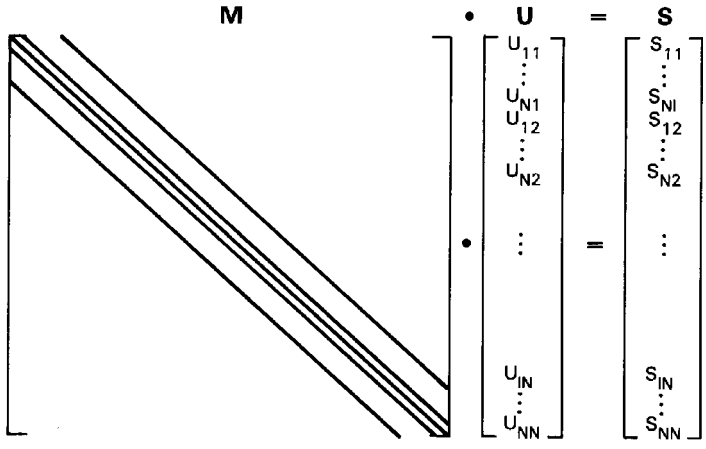

Figure 10.13. Generic form of a pentadiagonal matrix. The slanted solid lines indicate nonzero coefficients and the blank areas assume zero values in the matrix **M**.

In two dimensions the generalization of equation (10–4.3) is

$$\frac{d\rho}{dt} = D\frac{d^2\rho}{dx^2} + D\frac{d^2\rho}{dy^2} + s(x, y, t). \tag{10–4.8}$$

With $\Delta \equiv \Delta x = \Delta y$, a simple finite-difference approximation to equation (10–4.3) is

$$\begin{aligned}\frac{\rho^n - \rho^{n-1}}{\Delta t} = {} & \frac{D\theta}{\Delta^2}[(\rho_{i+1,j} - 2\rho_{ij} + \rho_{i-1,j}) + (\rho_{i,j+1} - 2\rho_{ij} + \rho_{i,j-1})]^n \\ & + \frac{D(\theta - 1)}{\Delta^2} \times [(\rho_{i+1,j} - 2\rho_{ij} + \rho_{i-1,j}) \\ & + (\rho_{i,j+1} - 2\rho_{ij} + \rho_{i,j-1})]^{n-1}.\end{aligned} \tag{10–4.9}$$

Now $\mathbf{M}^n$ and $\mathbf{M}^{n-1}$ are no longer tridiagonal, but have the pentadiagonal form shown schematically in Figure 10.13. The four off-diagonal rows of coefficients are not contiguous, although the matrix is still sparse. In fact, rather general partial differential equations can be approximated in this form as long as they can be approximated using the five-point template depicted in Figure 7.7a. Equation (10–4.1) encompasses all of these cases, but also disguises the underlying structure which can usually be exploited to optimize the numerical solution.

10–4.2. Exact Solution of the Matrix Equation

There are relatively fast iterative methods that produce reasonable solutions. As in all iterations, however, the rate of convergence and the pathological cases determine how useful the method really is. Most of the general methods that give "exact" solutions for $\mathbf{M}^{-1}$, and that do not take advantage of any special properties of the matrix, are based on *Gaussian elimination* and scale as N^3. The basic idea of Gaussian elimination is shown when equation (10–4.1) is written in component form

$$\sum_j m_{ij} u_j = s_i, \qquad i = 1, \ldots, N. \tag{10–4.10}$$

The first equation can be used to define u_1 in terms of the other $(N-1)$ variables. Then, using this equation, u_1 is successively eliminated from the remaining $(N-1)$ equations by multiplying each of the $(N-1)$ other equations by m_{11}/m_{i1} and subtracting the first equation. This leaves the partially reduced matrix equation

$$\begin{aligned} m_{11}u_1 + m_{12}u_2 + \cdots + m_{1N}u_N &= s_1 \\ m'_{22}u_2 + \cdots + m'_{2N}u_N &= s'_2 \\ m'_{32}u_2 + \cdots + m'_{3N}u_N &= s'_3 \\ \vdots \qquad\qquad \vdots \qquad &= \vdots \\ m'_{N2}u_2 + \cdots + m'_{NN}u_N &= s'_N. \end{aligned} \tag{10–4.11}$$

The same process eliminates all of the other variables from successively smaller sets of equations, ending up with an upper triangular matrix equation with only nonzero elements below the diagonal,

$$\begin{aligned} m_{11}u_1 + m_{12}u_2 + m_{13}u_3 + \cdots + m_{1N}u_N &= s_1 \\ m'_{22}u_2 + m'_{23}u_3 + \cdots + m'_{2N}u_N &= s'_2 \\ m''_{33}u_3 + \cdots + m''_{3N}u_N &= s''_3 \\ \vdots \\ m^{N-1}_{NN}u_N &= s^{(N-1)}_N. \end{aligned} \tag{10–4.12}$$

This process requires several times $\mathcal{O}(N^3)$ operations to eliminate the elements below the diagonal, and great care to control errors that can occur because of large differences in the matrix elements. The numerical process that completes the solution is called *back substitution*. First solve for u_N from the last of equation (10–4.12). Then use the known value of u_N in the next to last equation and solve for u_{N-1}. Repeat this substitution procedure until all values including u_1 are known. Back substitution requires on the order of N^2 operations.

Because Gaussian elimination scales as N^3, it is too expensive to be used repeatedly in an evolving simulation. We need a faster method, where "fast" generally means scaling approximately as $N \ln N$.

10–4.3. Tridiagonal Matrices

Suppose that the matrix is simpler, such that there is a set of equations of tridiagonal form that can be written as

$$\alpha_j u_{j-1} + \beta_j u_j + \gamma_j u_{j+1} = s_j, \tag{10–4.13}$$

for $1 < j < N$ with the appropriate boundary conditions at u_1 and u_N. Here the problems are usually one dimensional, so that $N = N_x$. Given u_j, we want a relation that gives us u_{j+1}. All easily solvable tridiagonal systems replace u_o and u_{N+1} in equation (10-4.13) with known values in terms of the bounding values of u in cells 1, 2, $N-1$, and N. For example,

$$u_o = s_o + \alpha_o u_1 + \beta_o u_2 + \gamma_o u_N, \tag{10–4.14}$$

and

$$u_{N+1} = s_{N+1} + \alpha_{N+1}u_1 + \beta_{N+1}u_{N-1} + \gamma_{N+1}u_N, \tag{10–4.15}$$

can be used to define u_o and u_{N+1} in equation (10–4.13). When the α_{N+1} and γ_o terms are nonzero, equations (10–4.14) and (10–4.15) correspond to periodic boundary conditions. Then Gaussian elimination can be used in $\mathcal{O}(N)$ operations without filling the entire upper triangular matrix.

A simple scalar approach to solving the tridiagonal matrix equation (10–4.13) results in recursion relations. Consider a simple recursion for u_{j+1},

$$u_{j+1} = x_j u_j + y_j, \tag{10–4.16}$$

that can be substituted into equation (10–4.13). Rearranging terms gives

$$u_j = -\frac{\alpha_j}{\gamma_j x_j + \beta_j} u_{j-1} + \frac{s_j - \gamma_j y_j}{\gamma_j x_j + \beta_j}. \tag{10–4.17}$$

We can now equate the terms in equations (10–4.16) and (10–4.17) to give

$$\begin{aligned} x_{j-1} &= -\frac{\alpha_j}{\gamma_j x_j + \beta_j} \\ y_{j-1} &= \frac{s_j - \gamma_j y_j}{\gamma_j x_j + \beta_j}, \end{aligned} \tag{10–4.18}$$

for all of the values of $\{x_j\}$ and $\{y_j\}$.

The procedure is then to start at N, and work down to $j = 1$ to find the values of x_j and y_j. Then to go from $j = 1$ to $j = N$ using equation (10–4.16) to produce the solution $\{u_j\}$. The result is a double-recursion relation. It is essentially the same algorithm as obtained by Gaussian elimination and is generally accurate, stable, and scales as N. The operation count is minimal. It is not easily vectorized or set up for parallel processing unless a number of similar but separate tridiagonal systems can be solved simultaneously.

The $N \ln N$ folding algorithms for solving very large tridiagonal matrices are all quite similar to each other, and somewhat more complex than the double-sweep scalar recursion algorithm just given. The equations relating u_j for even j to u_{j+1} and u_{j-1} are used to eliminate u_j from the odd j equations, giving half as many equations. New, composite coefficients relate alternate unknowns. This new system is still tridiagonal and thus can be folded again. When the repeatedly folded system becomes short enough, the scalar algorithm is used to complete the solution of the short system. Using the (now known) variables with odd j at each stage of the unfolding, the even j unknowns can be found. Algorithms for all of these cases and corresponding test problems have been assembled by Boris (1976). These $N \ln N$ cyclic reduction algorithms vectorize fairly well and can be used for parallel processing.

As multidimensional problems become larger with many tridiagonal systems to be solved simultaneously, the most efficient implementation is still the scalar recursion algorithm solved for a number of lines in parallel. This makes efficient use of the cache-based architecture for highly parallel computing. The possibility also exists of using large tridiagonal systems in iterative solutions to solve multidimensional elliptic problems with

variable coefficients. For example, by tracing a "fractal" path through all the nodes in a multidimensional grid, two of the neighbors of each node can be treated implicitly while the other two (or four) nodes can be included in the five-point (or seven-point) stencil explicitly. Thus the entire grid is coupled each iteration by a tridiagonal solution involving all $N_c = N_x \times N_y \times N_z$ nodes.

10–4.4. Direct Solutions of the Poisson Equation

The simplest case of a multidimensional sparse-matrix problem with pentadiagonal form arises from a local five-point finite-difference approximation to the Poisson equation on a rectangular grid. Even when the cells are nonuniform in size and shape, we retain the regular sparse matrix structure as long as the grid is topologically rectangular and each vertex has four adjacent cells. By taking advantage of the fact that the nonzero elements of the matrix are localized in the diagonal and adjacent blocks, the pattern of the entries allows us to reduce the operation count to $\mathcal{O}(N^2)$ because most of the blocks are entirely zero. We would like algorithms for inverting the two-dimensional five-point Poisson operator in $\mathcal{O}(N \ln N)$ operations, where $N = N_x \times N_y$ is the number of distinct potential values sought. On a 100×100 mesh, an $\mathcal{O}(N \ln N)$ solution reduces the solution time by about a factor of 15.

General elliptic equations in two or three dimensions are difficult and expensive to solve, particularly when antisymmetric cross terms are comparable in magnitude to the symmetric terms. Symmetry in the form of the second-order derivatives in the operator,

$$\nabla^2 = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}, \tag{10–4.19}$$

makes possible several direct, fast solution algorithms that do not apply in the general case. Because of these symmetries, the eigenfunctions of equation (10–4.19) can often be written as the product of separate eigenfunctions along each of the grid directions. The result is that multidimensional problems can sometimes be separated into a set of one-dimensional problems, each of which is tridiagonal. Some direct methods are described briefly in the following paragraphs.

A number of well-tested packages exist to solve elliptic and Poisson equations in multidimensions. Before writing your own, check the software library on your computer system or ask what is available at a few major computer centers. There are a number of public-domain software libraries that can be accessed. The journal *Computer Physics Communications* (published by North-Holland) is devoted to documenting and disseminating this kind of software. There is also a report (Boris 1976) containing algorithms and programs for a number of optimized tridiagonal solvers.

Multiple Fourier Analysis

This method involves performing a complete Fourier analysis along each spatial dimension (Hockney 1970; Boris and Roberts 1969). In two dimensions, this requires on the order of $\mathcal{O}(N \ln N)$ arithmetic operations and can be accomplished in the same operation count as a number of one-dimensional Fourier transforms or as a single two-dimensional transform. One of the attractions of this method is the ability to use standard, highly optimized

fast-Fourier transform (FFT) algorithms. The FFT is one of the best-studied numerical algorithms and usually is one of the first to be optimized for a new computer system. Sometimes the double Fourier harmonics of the potential ϕ are also eigenfunctions of the two-point gradient and three- and five-point second-derivative difference operators. Then it may be possible to perform the Fourier decomposition through the finite-difference operators. This decomposes the problem into a large number of simple equations for the separate harmonics.

Multiple Fourier analysis algorithms scale as $N \ln N$ when the system size is suitably chosen and can be used for three-dimensional problems. Typically FFTs are readily available when the number of cells in the system is a power of two or four. FFTs are also efficient for all system lengths that can be expressed as the products of small prime integers. Because of their use in spectral and pseudospectral methods (see Chapter 8), the best place to learn how to use FFTs under nonideal situations is in that literature. To use FFTs in a relatively straightforward way, a uniformly spaced grid and constant coefficients in the elliptic or Poisson equations are generally necessary. Periodic, symmetric (reflecting), and antisymmetric boundary conditions are also straightforward to apply.

The operation count using FFTs in one dimension is $N \ln N$. This is somewhat more than the $\mathcal{O}(N)$ scaling of tridiagonal algorithms but about the same as folding or cyclic reduction algorithms. Fourier transform algorithms are generally more accurate when they can be used because each harmonic contains information from everywhere else in the grid. Higher-order errors introduced by finite-difference derivative operators elsewhere in the calculation, for example, by differencing the velocity potential to obtain the velocity, can be corrected by adjusting the harmonic amplitudes. Birdsall and Langdon (1985) have reviewed these considerations in the context of plasma simulation.

Fourier Analysis Decoupling Tridiagonal Systems

This hybrid method involves Fourier analysis along one (or two) spatial directions to decouple the Fourier harmonics. It gives separate tridiagonal matrix equations in the transverse direction for each of the harmonics. This technique was introduced by Hockney (1970) who used a cyclic reduction (folding) technique on the tridiagonal equations. In two dimensions, this requires on the order of $N_x N_y \ln N_x N_y$ arithmetic operations. It is a very fast method and is applicable to axisymmetric and some other more complex geometries because the Fourier analysis is only done in one dimension.

A variant uses a scalar double-sweep algorithm for separated tridiagonal equations. Parallelization is straightforward because a number of tridiagonal systems of identical length must be solved at once. In three dimensions these hybrid techniques can be used if Fourier transforms in two directions can decouple the problem into a system of one-dimensional tridiagonal matrices.

Double Cyclic Reduction

This method involves cyclic reduction along each spatial dimension and was introduced by Buneman (1969). When the coefficients in each of the two spatial directions are constant, an analytic method eliminates half of the equations in each row using the basic five-point difference formulas to remove the alternate variables from the equations. Although cyclic

reduction can be used on tridiagonal matrices with rather general coefficients, the need to perform the reduction procedure in two directions at once appreciably constrains the coefficients for which the method will work.

Cyclic reduction is used repeatedly until the matrix is small enough to solve directly. The boundary and grid limitations are roughly similar to those needed to make Fourier transforms useful. The resulting optimized program can be faster because the cyclic reduction technique is somewhat more efficient than Fourier transforms. The double cyclic reduction method, however, lacks the flexibility of FFTs to introduce higher-order corrections to the finite-difference operators by modifying the Fourier coefficients.

10–4.5. Iterative Solutions of Elliptic Equations

In a few special cases, direct solutions of a complete elliptic system are possible in order $N_c \ln N_c$ operations, where N_c is $N_x \times N_y$ cells in two dimensions and $N_x \times N_y \times N_z$ in three dimensions. These special cases do not include problems with more general coefficients that arise, for example, from variable rectilinear gridding, a temperature-dependent diffusion coefficient, or a variable sound speed. It takes at least of order N_c^2 operations to solve equation (10–4.1) using Gaussian elimination or an equivalent method. For the large matrices of interest here, this operation count is not acceptable. Therefore we now consider iterative methods to use when fast, direct methods cannot be used. Extensions and generalization of the discussion given here can be found in Potter (1973) and Varga (1962).

An iterative method starts with a guess at an initial value of $\mathbf{u}$, called $\mathbf{u}^{(o)}$. The algorithm proceeds through p iterations, arriving at $\mathbf{u}^{(p)}$, which should be an improved approximation. The iteration formula is

$$\mathbf{u}^{(p+1)} = \mathbf{P} \cdot \mathbf{u}^{(p)} + \mathbf{c}, \tag{10–4.20}$$

where $\mathbf{P}$ is the iteration matrix related to $\mathbf{M}^{-1}$ and $\mathbf{c}$ is related to $\mathbf{s}$ and previous approximate solutions. Each improved solution vector $\mathbf{u}^{(p+1)}$ may be obtained explicitly from $\mathbf{u}^{(p)}$. Sometimes improvement is obtained by modifying $\mathbf{P}$ at each iteration. The methods described below all fit into this format, except for the last, the alternating-direction implicit method, which is a two-step iterative procedure. The computational issues are how many iterations are needed and what is the best form of $\mathbf{P}$. Simply using a matrix multiplication operation as implied by equation (10–4.20) requires $\mathcal{O}(N^2)$ operations per iteration. The important point is that $\mathbf{P} \cdot \mathbf{u}^{(p)}$ should require fewer operations than $\mathcal{O}(N^2)$ when $\mathbf{P}$ is suitably chosen.

Here we illustrate the methods by applying them to the Poisson equation, rewritten in a more graphic notation for the five-point difference stencil,

$$u_c - \frac{1}{4}(u_n + u_s + u_e + u_n) = \frac{1}{4} s_c \Delta^2 \equiv w_c, \tag{10–4.21}$$

where the (i, j)th point on the computational mesh is denoted c and the surrounding points are denoted n, s, e, and w for north, south, east, and west, respectively, as shown

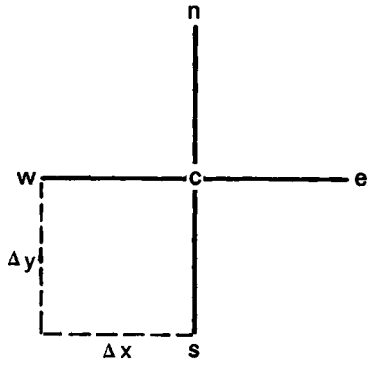

Figure 10.14. Schematic showing a particular grid point c in a computational mesh. This grid point is surrounded by north, south, east, and west points, designated n, s, e, and w, respectively.

in Figure 10.14. For illustrative purposes, $\Delta \equiv \Delta_x = \Delta_y = \Delta_z$. For iteration procedures applied to equation (10–4.21) we write

$$u_c^{(p+1)} = (1-\omega)u_c^{(p)} + \frac{1}{4}\omega\left(u_n^{(p)} + u_s^{(p)} + u_e^{(p)} + u_w^{(p)}\right) + \omega w_c, \tag{10–4.22}$$

where ω is an adjustable relaxation parameter. For convergence in two dimensions, $0 \leq \omega \leq 1$. When ω is zero, the new iteration is the same as the previous one. When ω is greater than one, the solution is being extrapolated beyond the current iteration.

Jacobi Iteration

This is equation (10–4.22) with $\omega = 1$, equivalent to averaging the four adjacent nodes to find the next iteration in Laplace's equation. The method converges slowly and is not commonly used.

Gauss-Seidel Iteration

The trick in this method is to note that each value of $\mathbf{u}^{(p+1)}$ is obtained in sequence. For example, when the central node c is reached in Poisson's equation, we will already have two of the adjacent values $u_s^{(p+1)}$ and $u_w^{(p+1)}$ and can use them as improved terms in the iteration. If $\omega = 1$, then

$$u_c^{(p+1)} = \frac{1}{4}\left(u_n^{(p)} + u_s^{(p+1)} + u_e^{(p)} + u_w^{(p+1)}\right) + w_c. \tag{10–4.23}$$

This method converges somewhat faster than the Jacobi iteration. It is, however, really only suitable for scalar computation because one computation must be completed before another can be started, that is, u_c^{p+1} depends on u_w^{p+1}.

The Successive Over-Relaxation (SOR) Method

This method uses the Gauss-Seidel formulation but optimizes the value of ω. For example, for the Poisson problem with large values of N_x and N_y, the best value of ω, denoted ω_b, is

$$\omega_b = \frac{2}{1+\sqrt{1-\mu_m^2}}, \tag{10–4.24}$$

where

$$\mu_m = \frac{1}{2}\cos\frac{\pi}{N_x} + \frac{1}{2}\cos\frac{\pi}{N_y}. \tag{10–4.25}$$

Therefore

$$\omega_b \approx \frac{2}{1 + \pi\sqrt{\frac{1}{2N_x^2} + \frac{1}{2N_y^2}}}, \tag{10–4.26}$$

which goes to the value $\omega = 2$ in the limit of an infinitely fine mesh. This method converges faster than the other methods described so far. The convergence is still slow in the early stages of iteration, because the error can only propagate away from a cell at a rate of one cell per iteration. Further, the error during these stages can even grow from step to step.

A variation on this is a "double" or "cyclic" iteration. This takes advantage of the fact that variables at even points are coupled directly only to variables at odd points, and vice versa. The idea is to improve all variables on even points first, and then use these to improve all the variables on odd points. Sometimes this is called "red-black" ordering, recalling the red and black squares on a chess board (see, for example, [Adams and Jordan 1984]). This iteration also generalizes to three dimensions.

Certainly SOR is the simplest algorithm. It is often best to use the SOR method to test various portions of a composite model in the development phases of large, complex models. The price for this conceptual and programming simplicity is slow convergence and high cost.

It is possible to extrapolate the starting iteration $\mathbf{u}^{(o)}$ in equation (10–4.20) from the previous two timesteps to get a better starting value for the SOR iteration. This extrapolation generally works better than might be expected because it is most accurate at long wavelengths where the iteration converges most slowly. Relatively large errors introduced by the extrapolation in the short-wavelength components of the solution decay away quickly while the starting error in the long wavelengths is often orders of magnitude smaller. The result, in terms of time advancement from one timestep to the next, is that the starting error in the longer wavelengths is much smaller in the initial guess and many fewer iterations are generally required to obtain a given accuracy.

Cyclic Chebyshev Method

This is similar to the previous method, except that the early stages of convergence are improved by varying ω from step to step using a specific sequence $\{\omega^{(p)}\}$. For the Poisson equation problem we are considering,

$$\begin{aligned}
\omega^{(0)} &= 1 \\
\omega^{(1)} &= \frac{1}{1 - \frac{1}{2}\mu_m^2} \\
&\vdots \qquad\qquad \vdots \\
\omega^{(p)} &= \frac{1}{1 - \frac{1}{4}\mu_m^2\,\omega^{(p-1)}},
\end{aligned} \tag{10–4.27}$$

where μ_m is defined in equation (10–4.25). Note that ω increases with each step, and in the limit, $\omega(\infty) = \omega_b$ from equation (10–4.24).

Alternating-Direction Implicit (ADI) Method

This method involves successive iterated implicit steps in each spatial direction. Each step involves the solution of a number of linear tridiagonal systems. For the Poisson equation in two dimensions, the method may be written

$$\begin{aligned} u_c^{(p+\frac{1}{2})} - u_c^{(p)} &= \frac{1}{2} w_c \left[(u_n - 2u_c + u_s)^{(p+\frac{1}{2})} + (u_e - 2u_c + u_w)^{(p)} \right] \\ u_c^{(p+1)} - u_c^{(p+\frac{1}{2})} &= \frac{1}{2} w_c \left[(u_n - 2u_c + u_s)^{(p+\frac{1}{2})} + (u_e - 2u_c + u_w)^{(p+1)} \right]. \end{aligned} \tag{10–4.28}$$

Extensions to three dimension are described in Richtmyer and Morton (1967).

The block implicit approach, applied in an ADI framework, is one way to calculate subsonic multidimensional flows (see, for example, Section 9–3; also Lindemuth and Killeen [1973]; Briley and McDonald [1975, 1977, 1980]; McDonald and Briley [1975]; Beam and Warming [1976, 1978]; McDonald [1979]). In this method, calculations are performed implicitly for one direction using a tridiagonal solver with the fluid properties from the other direction defined explicitly using the values of the physical variables from the previous timestep or iteration. The directions are then reversed and the ADI procedure is repeated. A number of iterations are required to couple the whole calculation accurately. When additional physical processes, such as chemical reactions or molecular diffusion, are included in the solution, this type of approach becomes complicated as the matrices that form each block become large.

The iterative methods described above all use the structure of the finite-difference grid and the sparsity of the matrix to reduce the operation count. Each iteration requires on the order of $N = N_c$ operations, where N_c is the number of computational cells, but at least $\max(N_x, N_y, N_z)$ iterations are required to share information over the entire grid. In two dimensions, a good approximate solution is obtained in $\mathcal{O}(N_c^{3/2})$ operations, and in three dimensions the scaling is $\mathcal{O}(N_c^{4/3})$ although N_c itself is usually very large. Even though these approximate scaling laws are an improvement over the ${N_c}^2$ scaling, $N_c \ln N_c$ scaling is much better if it can be obtained.

Incomplete Cholesky – Conjugate Gradient, ICCG Iteration

The *Cholesky* method is a form of Gaussian elimination for *symmetric* linear systems. The *conjugate gradient* method is an iterative method that uses successive approximations to reach the final solution. These two methods are often combined in the *incomplete Cholesky – conjugate gradient* (*ICCG*) method to solve systems of linear equations for physical problems in which the matrix is positive definite and nearly symmetric. Good descriptions of the ICCG method are given by Kershaw (1978) and Meijerink and van der Vorst (1977).

The conjugate gradient method is iterative and involves successive approximations, each of which is a projection of the solution on a progressively larger subspace of the vector space in which the solution lies. Each successive iteration searches for a correction to the previous iteration in a direction normal to the subspace of the most recent approximation. This

search continues until the entire vector space is searched. When the matrix is symmetric and positive definite, convergence to the exact solution is assured in N iterations. Unfortunately, N is quite large and, when the matrix is not symmetric, convergence is not assured.

Conjugate gradient methods are often improved with some kind of "preconditioning" of the matrix. If convergence is only required to some finite accuracy, then it is possible to precondition the matrix so the solution converges to within that finite value in considerably fewer than N iterations. The incomplete Cholesky method is one way to do this.

The ICCG method works best for physical problems in which the matrix is almost symmetric. This is especially true for problems in which viscous or diffusive effects are dominant. These include slow viscous fluid flows, magnetohydrodynamic problems in which resistivity or viscosity dominate, or Fokker-Planck equations in which the collision operator is a diffusion term. The method is also a good option for solving Poisson equations with complicated grids or geometry, in which the matrix is very large and symmetric but of complicated structure.

10–4.6. Multigrid Iteration Methods

The multigrid method, described next, and the ICCG method are now being used as alternatives to the simpler, but relatively expensive SOR and ADI iterations. As these methods become as well tuned and exhaustively tested, they will approach matrix methods in general availability and ease of use. They are also similar to matrix-inversion techniques in that multigrid and ICCG methods use a relatively large amount of storage. Many times as much scratch memory is often required as there are cells in the two- or three-dimensional finite-difference mesh being solved. Good introductions to multigrid methods are given in the books by Briggs (1987) and Wessling (1992).

Multigrid methods combine direct and iterative methods. As such, they combine the advantages of iterative methods, in terms of the rapid relaxation of fine-scale errors, and direct methods, in terms of the reduced effort required as the number of unknowns is reduced. During the last two decades, there has been considerable effort devoted to multigrid methods for solving many types of numerical problems. The general principles of these methods apply equally well to finite-difference and finite-element approaches for solving partial differential equations. We recommend the book edited by Hackbusch and Trottenberg (1983) and the articles by Brandt (1977, 1981). The basic idea of multigrid methods was originally presented by Fedorenko (1961).

The essence of the multigrid approach is to set up a discretization of the computational domain, and to define a sequence of coarser, perhaps nested, auxiliary discretizations in this domain. The solutions on these coarser (less expensive) grids can be used to approximate, and hasten the convergence of, the solution on the finer grids. The solution on any grid can be improved by combining relaxation iterations on that grid, which smooth the fine-scale errors, and correction iterations using the coarser grids, which reduce the large-scale errors. The coarser grids have many fewer unknowns, and so require far less computational effort than the iterations on the finest grid. The final result can be a substantial savings in the work required to solve the problem and has an $N \ln N$ scaling.

Consider multigrid algorithms for elliptic problems. These have been proposed and analyzed by a number of authors, for example, Brandt (1977), Bank and Dupont (1981),

and Douglas (1984). Douglas found an efficient multigrid algorithm for elliptic problems and this was subsequently converted to vectorized form for high-performance computing by DeVore (1984).

Suppose we have an elliptic equation to solve in the (x, y) plane with an $N_c = N_x \times N_y$ grid with uniform spacing h in each direction. After finite differencing the differential equation and incorporating the boundary conditions, the problem is reduced to solving the matrix equation,

$$\mathbf{M}^h \cdot \mathbf{u}^h = \mathbf{s}^h, \tag{10–4.29}$$

where $\mathbf{M}^h$ is the matrix of coefficients, $\mathbf{u}^h$ is the desired solution, and $\mathbf{s}^h$ is the inhomogeneous source vector.

We now describe the basic elements of a multigrid algorithm for solving this elliptic problem. Choose N_x and N_y to be even, and lay two uniform grids on the rectangle, one with $N_x \times N_y$ spacing h, and the other $(N_x/2) \times (N_y/2)$ with mesh spacing $2h$. Equation (10–4.29) holds on the fine grid, and on the coarse grid

$$\mathbf{M}^{2h} \cdot \mathbf{u}^{2h} = \mathbf{s}^{2h}. \tag{10–4.30}$$

We begin by solving equation (10–4.30) using a direct method. Because the number of unknowns is smaller by a factor of four, this requires a factor of eight less work than the same task on the fine grid. We next interpolate the solution onto the fine grid. The result has fine-scale errors due to the lack of fine-scale information, in addition to any errors introduced by the interpolation process. We therefore do a small number of relaxation iterations on the new solution to smooth the fine-scale errors, and this process produces the first approximation, $\mathbf{u}^{h(1)}$. We can improve the approximation and find $\mathbf{u}^{h(2)}$ by solving a correction problem based on the residual vector. This step requires a projection step in which values from the fine grid are projected onto the coarse grid, usually by summing or averaging.

This process describes a 2-level, 2-cycle algorithm. It could be extended to use additional coarse grids with spacings $4h$, $8h$, and so on, or to solve additional correction cycles, or both, leading to an $N \ln N$ algorithm. Various multilevel algorithms have been proposed by Brandt (1977) and Douglas (1982).

10–4.7. Summary

Transforming the partial differential equations of reactive flow into a finite number of algebraic relations produces a set of equations that can be written, manipulated, and analyzed in matrix form. In some situations, specifically in an explicit formulation, these equations can be solved directly with minimal effort. Very often, however, implicit solution algorithms are needed. In implicit time-dependent problems, a different elliptic equation must be solved at each timestep.

Iterative techniques such as SOR and ADI methods are useful and easy to program but are computationally expensive relative to faster, direct solution methods. ADI methods are intermediate in complexity between full implicit matrix methods and explicit SOR techniques. One of the attractions of the expensive procedure of simply solving the huge

matrices is the possibility of using a well-tested, documented, optimized package that will assure the correct solution to required accuracy.

Fast direct methods have been considered extensively by Hockney (1970) and Hockney and Eastwood (1981) and are very attractive when they can be used. For these methods, the cost in floating-point operations in multidimensional applications scales as $\mathcal{O}(N_c \ln N_c)$. When the finite-difference coefficients are constant in any direction – that is, in periodic or reflecting Cartesian geometries or the azimuthal direction in cylindrical coordinates – Fourier-transform and cyclic-reduction (folding) techniques can be used to decouple the solution into smaller distinct parts, thus reducing the effective dimensionality of the problem. When a problem is reduced to one dimension, a direct solution method exists using fast tridiagonal algorithms that also scale as $N_c \ln N_c$.

When the coefficients are variable in the elliptic equation, whether for physical or geometric reasons, the $N_c \times N_c$ matrix is initially very sparse but rapidly fills in. Direct solutions are still possible but generally much more expensive, scaling as $\mathcal{O}({N_c}^3)$ or $\mathcal{O}({N_c}^2)$. In these situations, some form of iteration using an approximate inverse is usually used to speed up the solution. The ICCG and multigrid methods and a number of other variants all recover the $N_c \ln N_c$ scaling of a fast direct method, but generally with a significantly larger numerical factor in front. The efficiency of these iterative algorithms is higher for a required accuracy than using methods based on ADI or SOR iteration, but many iterations are still needed in some situations. A direct method discussed by Madala (1978) is based on an error-sweepout technique and this has been used for difficult problems in weather modeling. Unfortunately no generalization of the efficient direct solutions of Buneman (1969) or Hockney (1970) to the general-coefficient elliptic problem on even a rectilinear grid has been discovered.

REFERENCES

Adams, L.M., and H.F. Jordan. 1984. *Is SOR color-blind?* ICASE report no. 84-14. Hampton, Va.: NASA Langley Research Center.

Amsden, A.A. 1966. *The particle-in-cell method for the calculation of the dynamics of compressible fluids*. LA-3466. Los Alamos, N. Mex.: Los Alamos Scientific Laboratory.

Bank, R.E., and T. Dupont. 1981. An optimal order process for solving elliptic finite element equations. *Mathematics of Computation* 36:35–51.

Barr, P.K., and W.T. Ashurst. 1984. *An interface scheme for turbulent flame propagation*. SAND82-8773. Livermore, Calif.: Sandia National Laboratory.

Bayliss, A., and E. Turkel. 1980. Radiation boundary conditions for wavelike equation. *Communications on Pure and Applied Mathematics* 33:707–725.

———. 1982. Far field boundary conditions for compressible flows. *Journal of Computational Physics* 48:182–199.

Beam, R.M., and R.F. Warming. 1976. An implicit finite-difference algorithm for hyperbolic systems in conservation-law form. *Journal of Computational Physics* 22:87–110.

———. 1978. An implicit scheme for the compressible Navier-Stokes equations. *AIAA Journal* 16:393–402.

Berenger, J.P. 1994. A perfectly matched layer for the absorption of electromagnetic waves. *Journal of Computational Physics* 114:185–200.

———. 1996. Three-dimensional perfectly matched layer for the absorption of electromagnetic waves. *Journal of Computational Physics* 127:363–379.

Birdsall, C.K., and A.B. Langdon. 1985. *Plasma physics via computer simulation*. New York: McGraw-Hill.

Boris, J.P. 1976. *Vectorized tridiagonal solvers*. NRL memorandum report no. 3408. Washington, D.C.: Naval Research Laboratory.

Boris, J.P., and K.V. Roberts. 1969. The optimization of particle calculations in two and three dimensions. *Journal of Computational Physics* 4:552–571.

Brandt, A. 1977. Multi-level adaptive solutions to boundary-value problems. *Mathematics of Computation* 31:333–390.

———. 1981. Multigrid solvers on parallel computers. In *Elliptic problem solvers*, ed. M.H. Shultz, 39–83. New York: Academic Press.

Bray, K.N.C., and Peters N. 1994. Laminar flamelets in turbulent flames. In *Turbulent reacting flows*, eds. P.A. Libby and F.A. Williams, 63–114. San Diego: Academic Press.

Brebbia, C.A., and J. Domingues. 1992. *Boundary elements*. New York: Computational Mechanics Publications, Southhampton and McGraw-Hill.

Briggs, W.L. 1987. *A multigrid tutorial*. Philadelphia: SIAM.

Briley, W.R., and H. McDonald. 1975. Solution of the three-dimensional compressible Navier Stokes equations by an implicit technique. In *Proceedings Fourth International Conference on Numerical Methods in Fluid Dynamics*, ed. R.D. Richtmyer, 105–110. New York: Springer-Verlag.

———. 1977. Solution of the multi-dimensional compressible Navier-Stokes equations by a generalized implicit method. *Journal of Computational Physics* 24:372–397.

———. 1980. On the structure and use of linearized block implicit schemes. *Journal of Computational Physics* 34:54–73.

Buneman, O. 1969. *A compact non-iterative Poisson solver*. Stanford University Institute for Plasma Research, report no. 294. Palo Alto, Calif.: Stanford University.

Chorin, A.J. 1980. Flame advection and propagation algorithms. *Journal of Computational Physics* 35:1–11.

DeVore, C.R. 1984. *Vectorization and implementation of an efficient multigrid algorithm for the solution of elliptic partial differential equations*. NRL memorandum report 5504. Washington, D.C.: Naval Research Laboratory.

Dongarra, J.J., I.S. Duff, D.C. Sorensen, and H.A. van der Vorst. 1998. *Numerical linear algebra for high-performance computers*. Philadelphia: SIAM.

Douglas, C.C. 1982. *Multi-grid algorithms for elliptic boundary-value problems*. Yale University technical report no. 223. New Haven, Conn.: Yale University.

———. 1984. Multi-grid algorithms with applications to elliptic boundary-value problems, *SIAM Journal of Numerical Analysis* 21:236–254.

Dutt, P. 1988. Stable boundary conditions and difference schemes for Navier-Stokes equations. *SIAM Journal of Numerical Analysis* 25(2):245–267.

Engquist, B., and A. Majda. 1977. Absorbing boundary conditions for the numerical simulation of waves. *Math. Comp.* 31:629–651.

———. 1979. Radiation boundary conditions for acoustic and elastic wave calculations. *Comm. Pure Appl. Math.* 32:312–358.

Farouk, B., E.S. Oran, and T. Fusegi. 2000. Numerical study of thermoacoustic waves in an enclosure. *Physics of Fluids* 12:1052–1061.

Fedorenko, R.P. 1961. A relaxation method for solving elliptic difference equations. *Z. Vycisl. Mat. i. Mat. Fiz.* 1:922–927.

Givoli, D. 1991. Non-reflecting boundary conditions. *Journal of Computational Physics* 94:1–29.

Grinstein, F.F. 1994. Open boundary conditions in the simulation of subsonic turbulent shear flows. *Journal of Computational Physics* 115:43–55.

Grosch, C.E., and S.A. Orszag. 1977. Numerical solution of problems in unbounded regions: Coordinate transformations. *Journal of Computational Physics* 25:273–295.

Gustafsson, B., and H.-O. Kreiss. 1979. Boundary conditions for time dependent problems with an artificial boundary. *Journal of Computational Physics* 30:333–351.

Gustafsson, B., and J. Oliger. 1982. Stable boundary approximations for implicit time discretizations for gas dynamics. *SIAM Journal on Scientific and Statistical Computing* 3(4):408–421.

Gustafsson, B., and A. Sundström. 1978. Incomplete parabolic problems in fluid-dynamics. *SIAM Journal of Applied Mathematics* 35(2):343–357.

Hackbusch, W., and U. Trottenberg. 1983. *Multigrid methods*. Lecture notes in mathematics, nr. 960. New York: Springer-Verlag.

Hagstrom, T., and S.I. Hariharan. 1988. Accurate boundary conditions for exterior problems in gas dynamics. *Mathematics of Computation* 51:581–597.

Harlow, F.H. 1955. *A machine calculation method for hydrodynamic problems*. LAMS-1956. Los Alamos, N. Mex.: Los Alamos Scientific Laboratory.

Harlow, F.H., and J.F. Welch. 1965. Numerical calculation of time-dependent viscous incompressible flow of fluid with free surface. *Physics of Fluids* 8:2182–2189.

Higden, R.L. 1986. Initial-boundary value problems for linear hyperbolic systems. *SIAM Review* 28:177–217.

Hirt, C.W., and B.D. Nichols. 1981. Volume of fluid (VOF) method for the dynamics of free boundaries. *Journal of Computational Physics* 39:201–225.

Hockney, R.W. 1970. The potential calculation and some applications. In *Methods in Computational Physics* 9, eds. B. Adler, S. Fernbach, and M. Rotenberg, 135–211. New York: Academic Press.

Hockney, R.W., and J.W. Eastwood. 1981. *Computer simulation using particles*. New York: McGraw-Hill.

Hyman, J.M. 1984. Numerical methods for tracking interfaces. *Physica* 12D:396–407.

Kaplan, C.R., G. Patnaik, and K. Kailasanath. 1998. Universal relationships in sooting methane-air diffusion flames. *Combustion Science and Technology* 131:39–65.

Kershaw, D.S. 1978. The incompete Cholesky – Conjugate gradient method for the solution of systems of linear equations. *Journal of Computational Physics* 26:43–65.

Kerstein, A.R., W.T. Ashurst, and F.A. Williams. 1988. Field equation for interface propagation in an unsteady homogeneous flow field. *Physical Review A* 37:2728–2731.

Khokhlov, A.M., E.S. Oran, and J.C. Wheeler. 1996. Scaling in buoyancy-driven turbulent premixed flames. *Combustion and Flame* 105:28–34.

Kreiss, H.–O. 1970. Initial boundary value problems for hyperbolic systems. *Communications in Pure and Applied Mathematics* 23:277–298.

Kythe, P.K. 1995. *Boundary element methods*. Boca Raton, Fla.: CRC Press.

Laskey, K.J., E.S. Oran, and J.P. Boris. 1987. *Approaches to resolving and tracking interfaces*. NRL memorandum report 5999, Washington DC: Naval Research Laboratory.

Lindemuth, I., and J. Killeen. 1973. Alternating direction implicit techniques for two-dimensional magnetohydrodynamic calculations. *Journal of Computational Physics* 13:181–208.

Madala, R.V. 1978. An efficient direct solver for separable and non-separable elliptic equations. *Monthly Weather Reviews* 106:1735–1741.

McDonald, H. 1979. Combustion modelling in two and three dimensions: Some numerical considerations. *Progress in Energy and Combustion Science* 5:97–122.

McDonald, H., and W.R. Briley. 1975. Three-dimensional supersonic flow of a viscous or inviscid gas. *Journal of Computational Physics* 19:150–178.

Meijerink, J.A., and H.A. van der Vorst. 1977. An iterative solution method for linear systems of which the coefficient matrix is a symmetric M-matrix. *Mathematics of Computation* 31:148–162.

Moretti, G., and M. Abbett. 1966. A time-dependent computational method for blunt body flows. *AIAA Journal* 4:2136–2141.

Morton, K.W., and D.F. Mayers. 1994. *Numerical solution of partial differential equations*. Cambridge, England: Cambridge Univ. Press.

Nakamura, S. 1991. *Applied numerical methods with software*. Englewood Cliffs, N.J.: Prentice Hall.

Noh, W.F., and P. Woodward. 1976. SLIC (simple line interface method). In *Proceedings of the Fifth International Conference on Numerical Methods in Fluid Dynamics*, Vol. 59, *Lecture Notes in Physics*, eds. A.I. van de Vooren and P.J. Zandbergen, 330–340. New York: Springer-Verlag.

Oliger, J., and A. Sundström. 1978. Theoretical and practical aspects of some initial boundary value problems in fluid dynamics. *SIAM Journal of Applied Mathematics* 35:419–446.

Osher, S., and J.A. Sethian. 1988. Fronts propagating with curvature-dependent speed: Algorithms based on Hamilton-Jacobi formulation. *Journal of Computational Physics* 79:12–49.

Ott, J.D. 1999. *The interaction of a laminar flame with its self-induced boundary layer in a rectangular channel*. PhD Thesis, Department of Aerospace Engineering, University of Maryland, College Park.

Poinsot, T.J., and S.K. Lele. 1992. Boundary conditions for the direct simulations of compressible viscous flows. *Journal of Computational Physics* 101:104–129.

Potter, D. 1973. *Computational physics*. New York: Wiley.

Pulliam, T.H. 1982. Characteristic boundary conditions for the Euler equations. In *Numerical Boundary Condition Procedures*, ed. P. Kutler. NASA Conference Publication 2201. Moffett Field, Calif.: NASA Ames Research Center.

Ralston, A. 1978. *A first course in numerical analysis*. New York: McGraw-Hill.

Richtmyer, R.D., and K.W. Morton. 1967. *Difference methods for initial-value problems*. New York: Interscience.

Rider, W.J., and D.B. Kothe. 1998. Reconstructing volume tracking. *Journal of Computational Physics* 141:112–152.

Roache, P.J. 1982. *Computational fluid dynamics*. Albuquerque, N. Mex.: Hermosa Publishers.

Rudman, M. 1997. Volume tracking methods for interfacial flow calculations. *International Journal for Numerical Methods in Fluids* 24:671–691.

Sethian, J.A. 1996. *Level set methods*. New York: Cambridge University Press.

Shyy, W., H.S. Udaykumar, M.M. Rao, and R.W. Smith. 1996. *Computational fluid dynamics with moving boundaries*. Washington, D.C.: Taylor and Francis.

Tam, C.K.W. 1998. Advances in numerical boundary conditions for computational aeroacoustics. *Journal of Computational Acoustics* 6:377–402.

Tam, C.K.W., and J.C. Webb. 1993. Dispersion-relation-preserving finite difference schemes for computational acoustics. *Journal of Computational Physics* 107:262–281.

Tannehill, J.C., D.A. Anderson, and R.H. Pletcher. 1997. *Computational fluid mechanics and heat transfer*. Washington, D.C.: Taylor and Francis.

Thompson, K.W. 1987. Time-dependent boundary conditions for hyperbolic systems. *Journal of Computational Physics* 89:1–24.

———. 1990. Time-dependent boundary conditions for hyperbolic systems, II. *Journal of Computational Physics* 68:439–461.

Varga, R.S. 1962. *Matrix iterative analysis*. Englewood Cliffs, N.J.: Prentice Hall.

Weber, Y.S., E.S. Oran, J.P. Boris, and J.D. Anderson, Jr. 1995. The numerical simulation of shock bifurcation near the end wall in a shock tube. *Physics of Fluids* 7:2475–2488.

Weber, Y.S., J.W. Weber, J.D. Anderson, and E.S. Oran. 1993. *The uniform boundary algorithm for supersonic and hypersonic flows*, In *parallel computational fluid dynamics '92*, eds. R.B. Pelz, A. Ecer, and J. Häuser, 395–406. New York: Elsevier.

Welch, J.E., F.H. Harlow, J.P. Shannon, and B.J. Daly. 1966. *The MAC method: A computing technique for solving viscous, incompressible, transient fluid flow problems involving free surfaces*. LA-3425. Los Alamos, N. Mex.: Los Alamos Scientific Laboratory.

Wessling, P. 1992. *An introduction to multigrid methods*. New York: Wiley.

Williams, F.A. 1985. Turbulent combustion. In *The mathematics of combustion*, ed. J. Buckmaster, 97–131. Philadelphia: SIAM.

Yee, H.C. 1981. *Numerical approximation of boundary conditions with applications to inviscid equations of gas dynamics*. NASA technical memorandum 81265. Moffett Field, Calif.: Ames Research Center.

11

Coupling Models of Reactive-Flow Processes

The previous chapters described techniques for solving the equations used to model different physical terms in the reactive-flow equations using algorithms that seemed most appropriate for each particular type of term. In each case, we identified those methods that would be best to combine and use in a reactive-flow program. This chapter delves into a fundamental issue in numerical simulations of reactive-flows: how to put all of this together in one computer model. How do we couple these separate algorithms in a way that is accurate enough and produces efficient yet flexible programs?

In a reacting flow, the different physical processes occur simultaneously, *not* separately or sequentially. For example, any temperature increase due to chemical reactions causes a local expansion of the gas *at the same time* the reactions are occurring, not some finite time later. There are at least two computational problems that result from this. First, the simulations must reproduce the correct physics of the interactions, even if it is not contained in the separate processes treated sequentially. Second, the coupling among parts of the equations representing different physical processes can be mathematically stiff. This is stiffness in the same sense discussed in Chapter 5, where some of the equations representing changes in densities of different reacting species may be mathematically stiff. The problem of coupling different processes becomes very serious if the system is characterized by multiple time and space scales.

The last section in Chapter 4, Section 4–6, gave a brief introduction to the coupling problem and highlighted the two main approaches, global-implicit methods and timestep-splitting methods. These are based on two very different philosophies, both of which need to be considered and understood in the context of complex, multiprocess reactive flows. In *timestep-splitting methods*, each type of term in the reactive-flow equations is treated separately and the results are then combined. This method is also called *operator splitting* or the *fractional-step method*, depending on the context. When it is applied to the different directions in a multidimensional process, such as to convection or diffusion, it is often called *direction splitting*. In global-implicit methods, also called *block-implicit methods*, all of the terms are considered together and the complete set of equations is linearized and solved. In addition, there are *hybrid coupling methods* that apply different approaches to selected groups of terms in the equations.

This chapter discusses these two main approaches and summarizes the strong and weak points of each. Then we describe how to build a complex reactive-flow model based primarily on timestep splitting. There are two related computational issues that pertain to using either of these approaches or even hybrid approaches. One issue concerns the physics of equilibrium and change when there are several interacting (competing) physical processes. The second relates to timestep control for the integration. Aspects of these two issues appear repeatedly in the material presented here. This chapter also presents a solution to a problem that has plagued numerical simulations of reactive flow: how to use timestep splitting to solve problems with multiple stiff processes.

The chapter ends with a discussion of some of the larger issues in numerical simulation of problems with extremely disparate time and space scales. A way of classifying the disparity in spatial and temporal scales is proposed, and then this classification is used to introduce a more advanced type of problem requiring a type of coupling called *intermittent embedding*. This could be used, for example, for situations in which several entirely different levels of solution methods, as listed in Table 1.1, are needed in the same problem.

11–1. Standard Approaches to Coupling Multiple Time Scales

The set of coupled, nonlinear partial differential equations describing a reactive-flow may be written in a general form as

$$\frac{\partial}{\partial t}\boldsymbol{\rho}(\mathbf{x}, t) = \mathbf{G}(\boldsymbol{\rho}, \nabla\boldsymbol{\rho}, \nabla\nabla\boldsymbol{\rho}, \mathbf{x}, t), \tag{11–1.1}$$

where $\boldsymbol{\rho}$ is now a vector, each component of which is a function of the time t and the position $\mathbf{x}$. The objective is to solve this set of equations, with the appropriate additional terms that describe the problem of interest, in a way that is general enough, modular enough, and accurate enough.

11–1.1. Global-Implicit Coupling

A global-implicit method solves equation (11–1.1) using fully implicit finite-difference formulas derived for the complete set of equations. This is sometimes called the block-implicit approach because of the mathematical form it takes in some applications involving a structured spatial grid. As the word "implicit" implies, the right side of equation (11–1.1) is formally evaluated using information at the advanced time. To do this, the nonlinear terms in $\mathbf{G}$ are linearized about the known solutions at the previous timestep to reduce the need for expensive iteration. This coupling approach is the direct generalization of the implicit algorithms presented throughout this book. In fact, when the finite-element approach described in Chapter 8 is used for coupling, it produces a type of global-implicit coupling that is determined by the specific finite-element formalism used for treating the spatial derivative.

Now write equation (11–1.1) in the form

$$\frac{\boldsymbol{\rho}^n(\mathbf{x}, t^n) - \boldsymbol{\rho}^{n-1}(\mathbf{x}, t^{n-1})}{\Delta t} = \mathbf{G}^n(\boldsymbol{\rho}^n, \nabla\boldsymbol{\rho}^n, \nabla\nabla\boldsymbol{\rho}^n, \mathbf{x}, t^n). \tag{11–1.2}$$

As before, superscripts n and $n-1$ represent two time levels separated by one timestep

Δt in the numerical solution. Equation (11–1.2) can be rewritten as

$$\boldsymbol{\rho}^n \approx \boldsymbol{\rho}^{n-1} + \Delta t\, \mathbf{G}^n, \tag{11–1.3}$$

where the arguments $\mathbf{x}$ and t are suppressed. Assume that changes in the values of $\boldsymbol{\rho}$ are small in one timestep, that is, that Δt is suitably small.

The difficulty with using equation (11–1.3) to find the new values of $\boldsymbol{\rho}$ is the implicit and generally nonlinear dependence of $\mathbf{G}$ on the new (unknown) values of $\boldsymbol{\rho}$. If the changes in $\boldsymbol{\rho}$ are small during a timestep, it is consistent to assume that the changes in $\mathbf{G}$ are also small. In this case we can approximate $\mathbf{G}^n$ by the first-order Taylor-series expansion

$$\mathbf{G}^n = \mathbf{G}^{n-1} + \Delta\boldsymbol{\rho}\, \frac{\partial \mathbf{G}^{n-1}}{\partial \boldsymbol{\rho}}. \tag{11–1.4}$$

The Jacobian, $\partial \mathbf{G}^{n-1}/\partial \boldsymbol{\rho}$, is a tensor when $\boldsymbol{\rho}$ is a vector of several variables. The Jacobian depends on the known values of $\boldsymbol{\rho}$ and hence can be evaluated explicitly. Defining $\Delta\boldsymbol{\rho}$ as the change in $\boldsymbol{\rho}$ from the old time to the new time,

$$\Delta\boldsymbol{\rho}^n \equiv \boldsymbol{\rho}^n - \boldsymbol{\rho}^{n-1} = \Delta t\, \mathbf{G}^n, \tag{11–1.5}$$

gives the global-implicit approximation for the new values of $\boldsymbol{\rho}$,

$$\Delta\boldsymbol{\rho}^n = \Delta t\, \mathbf{G}^{n-1}\left[\mathbf{1} - \Delta t\, \frac{\partial \mathbf{G}^{n-1}}{\partial \boldsymbol{\rho}}\right]^{-1}. \tag{11–1.6}$$

An implicit matrix equation results from this procedure, and a large and sometimes cumbersome evolution matrix, $1 - (\Delta t)\partial \mathbf{G}^{n-1}/\partial \boldsymbol{\rho}$, must be inverted in the general case.

Linear approximations to the convective derivatives are used to implement this formalism, generally preventing use of the more accurate nonlinear convection algorithms. A rigorously convergent nonlinear treatment also requires iterations of matrix inversions at each timestep. In one dimension, implicit coupling often requires solving a block tridiagonal matrix with M physical variables at N_x grid points. In this case, an $(MN_x \times MN_x)$ matrix must be inverted at each iteration of each timestep. The blocks on or adjacent to the matrix diagonal are $M \times M$ in size, so that the overall matrix is quite sparse. Nevertheless, a large amount of computational work goes into advancing the solution a single timestep. In this approach, the matrices are $(MN_xN_y \times MN_xN_y)$ in two dimensions. This computational load becomes even worse when higher-order approximations to the spatial derivatives are used.

When only a single process is involved, such as convection or diffusion, implicit methods can be relatively inexpensive. It is then possible to combine the equations to obtain an implicit elliptic equation, which is often much easier to solve than the general matrix equation given in equation (11–1.6). In global-implicit convection algorithms, as in linear explicit flow algorithms, first-order numerical damping (or diffusion) is required to maintain positivity. These implicit algorithms generally have the advantage of being numerically stable for long timesteps. Because the equations are linearized, the iterations generally include nonlinear terms that are partially or fully implicit.

An advantage of the global-implicit method is that all of the processes are considered simultaneously, so all physical interactions among processes are taken into account together. There should not, in principle, be any concern about synchronizing interactions and there is no question about the order in which processes occur. It seems to be a natural way to treat problems with multiple stiff processes. In complex kinetics problems with no spatial variation, the M fluid variables are the species number densities plus the temperature of the homogeneous volume of interest. The Gear method, discussed in Chapter 5, is an example of a global-implicit approach for integrating sets of coupled ordinary differential equations.

Generally the level of accuracy in an implicit Eulerian convection calculation with appreciable gradients is equivalent to that obtained using a monotonic convection algorithm that has about half the number of grid points in each spatial dimension. Thus the global-implicit approach is more expensive by about a factor of four in two dimensions and eight in three dimensions, even without adding any increased cost that would arise from appropriately reducing the timestep, as would be required for equivalent accuracy. Furthermore, in many situations the formal ability to take longer timesteps is of less practical use when rapid transient phenomena produce very small-scale spatial structures. When it is necessary to resolve these structures, short timesteps are required anyway.

11–1.2. Timestep Splitting

The second coupling method is timestep splitting, considered here in some detail. As noted earlier, this is also called operator splitting or the fractional-step method (Yanenko 1971). When it is applied to the different spatial directions in an algorithm for a multidimensional process, it is called direction splitting or the alternating-direction method (Douglas 1962). These terms are used interchangeably throughout this chapter. Other early related references include Strang (1968), Marchuk (1975), and more recently, Leveque and Oliger (1983).

The basic idea of timestep splitting is to evaluate independently the changes in the physical variables due to the different processes. Then these changes are combined in some way to obtain the composite change. The individual processes and the interactions among them may be treated by analytic, asymptotic, implicit, explicit, or other methods. An advantage of this approach is that it avoids many costly matrix operations and allows the best numerical method to be used for each type of term or process. A potential disadvantage is that the separate algorithms can be more complex and usually differ from term to term and even from one situation to another. The exact way the processes are coupled, therefore, may also vary for different types of problems.

The qualitative criterion for the validity of timestep splitting is that, during a timestep, the values of the physical variables must not change too quickly from any of the individual processes. This is also a requirement for global-implicit algorithms. To show how timestep splitting works, we can write equation (11–1.1) in the form

$$\frac{\partial}{\partial t}\boldsymbol{\rho}(\mathbf{x}, t) = \mathbf{G}_1 + \mathbf{G}_2 + \mathbf{G}_3 + \cdots + \mathbf{G}_M, \tag{11–1.7}$$

where $\mathbf{G}$ has been broken into M constituent processes,

$$\mathbf{G} = \mathbf{G}_1 + \mathbf{G}_2 + \mathbf{G}_3 + \cdots + \mathbf{G}_M. \tag{11–1.8}$$

Each function in the set $\{\mathbf{G}_i\}$ contributes some amount to the overall change in $\boldsymbol{\rho}$ during the numerical timestep. Thus

$$\left.\begin{aligned} \Delta\boldsymbol{\rho}_1^{n-1} &= \Delta t\, \mathbf{G}_1^{n-1} \\ \Delta\boldsymbol{\rho}_2^{n-1} &= \Delta t\, \mathbf{G}_2^{n-1} \\ &\vdots \quad \vdots \\ \Delta\boldsymbol{\rho}_M^{n-1} &= \Delta t\, \mathbf{G}_M^{n-1} \end{aligned}\right\}. \tag{11–1.9}$$

For example, $\mathbf{G}_1^{n-1}$ might be the chemical-kinetics or atomic-reaction terms, $\mathbf{G}_2^{n-1}$ the diffusion terms, $\mathbf{G}_3^{n-1}$ the thermal-conduction terms, and so on. The solution for the new values $\boldsymbol{\rho}^n$ is found by summing all of the partial contributions,

$$\boldsymbol{\rho}^n = \boldsymbol{\rho}^{n-1} + \sum_{m=1}^{M} \Delta\boldsymbol{\rho}_m^{n-1}. \tag{11–1.10}$$

If each of the processes is simulated individually, equation (11–1.10) gives a simple prescription for combining the results. This operator-splitting approach is discussed by Dean and Glowinski (1993) for situations in which the individual processes can be treated implicitly. Examples are given in which the separate operator integrations are combined using the Peaceman-Rachford scheme (1955) and the Douglas-Rachford scheme (1956).

If the eigenvalues of the Jacobian for each of the separate processes $\partial\mathbf{G}_i/\partial\rho$ are large, which means that these processes are mathematically stiff, explicit algorithms are stable for each of the processes individually only when the timestep is small enough. The timestep needed to satisfy all of the separate explicit stability conditions may be quite small. This is sometimes not a problem, as the gain achieved by avoiding matrix operations may more than compensate for the cost of a small timestep.

It is useful to distinguish between two types of situations in which the processes are stiff and require short timesteps. In one case, short timesteps are needed because rapid transients must be resolved. In another, short timesteps are needed because one or more of the Jacobian eigenvalues is large. In this second case, we are concerned with the existence of dynamic equilibria that appear as large terms nearly cancel each other. This latter case is the situation in which an individual $\mathbf{G}_i$, might be large, but the sum of several $\mathbf{G}_i$'s is small because of cancellation. We revisit this situation in Section 11–4.

In the second case, it is difficult to use straightforward timestep splitting. The origin of the difficulty is *not* because of a difference-of-large-numbers problem, but because of potential instabilities in the numerical solution of the equations associated with the large eigenvalues. It has been generally assumed that global-implicit coupling avoids this stiff-equilibrium problem. Nonetheless, instability can arise, even when using global-implicit methods, if the timestep is too large. When and how instabilities occur depend on how the equations are linearized.

11–1.3. Trade-offs in the Choice of Methods

The choice of a method to couple the physical processes is a decision that determines the overall structure of the reactive-flow model. Therefore, it is important to understand

the trade-offs between global-implicit and fractional-step coupling at the beginning of a project. For each approach, we consider stability, accuracy, and ease of use.

The global-implicit formalism can be adopted to treat almost any set of coupled partial differential equations. Stable solutions are guaranteed by added numerical damping that smoothes the solution. Because of this damping, it is sometimes difficult to determine if the answers have converged correctly. There are cases in the literature in which this approach has given a stable, yet the wrong, solution. Convergence of the computed solutions must be tested by increasing the spatial and temporal resolution, a process that may become particularly expensive when multidimensional implicit methods are used. In principle, global-implicit coupling maximizes the strain on computer resources and minimizes the strain on the modeler.

Timestep splitting becomes more accurate as the timestep becomes smaller, and it should be exact in the limit as the timestep goes to zero. Timestep splitting puts less strain on computational resources but demands more effort and thought on the part of the modeler. Stability is not guaranteed; it depends on the separate criteria of the algorithms used to represent the individual processes and the properties of their coupling. This can actually be an advantage because it is often easy to see when the answer is wrong; when the solutions fail, they degrade in catastrophic ways. For example, a typical signal of failure in a convection calculation is when the supposedly positive species densities become negative. Convergence is usually tested by increasing the spatial and temporal resolution. When some process or aspect of the problem uses asymptotic methods, however, situations arise for which reducing the timestep makes the answers less accurate. This failing of the model may also be catastrophic in the timestep-split calculation. For example, when an asymptotic calculation of a chemical-kinetics problem fails, mass is no longer conserved.

Timestep splitting encourages modular simulation models. Each module can be programmed using the best available technique for that process. If an improved technique becomes available, whether linear or nonlinear, it can often be incorporated without changing the entire structure of the model. Thus convection, diffusion, equation-of-state calculations, and species reaction kinetics are all in separate packages. If properly constructed, these separate modules can be combined with different representations of the other processes to solve different classes of problems. Particular care in testing the coupling between the software modules is required. This is both an advantage and a disadvantage for the timestep-splitting approach.

The pros and cons of using global-implicit methods versus timestep-splitting methods are roughly balanced with respect to accuracy of the solution. If an important physical timescale is not resolved, neither method can give detailed solutions for phenomena occurring on that scale. Similarly, to compute spatial gradients accurately, the gradients must be resolved by enough grid points in either type of calculation. The ability to easily incorporate monotone convection algorithms is a definite benefit of using timestep splitting. The fact that implementing timestep splitting demands more thought and effort is potentially counterbalanced by the fact that the resulting simulations generally are faster than those using global-implicit methods. More computations can be performed and results obtained more quickly, offsetting extra program-development time that may be needed. Often the reduction in computing time is substantial. For example, solving a set of reaction kinetics equations for M species requires inverting $M \times M$ matrices with approximately

M^3 operations. Other methods, such as asymptotic integration techniques, scale as M or M^2.

There are many circumstances in which it is advantageous to use a hybrid approach that combines the concepts of global-implicit and timestep-splitting methods. For example, when stiffness occurs through dynamic equilibria, it may be useful to couple the stiff processes implicitly. Then cancellation of the large terms takes place before the effects of the other terms are computed and the remainder of the reactive-flow problem can be done using explicit timestep splitting. This creates a type of hybrid model that requires knowing in advance that dynamic equilibria will occur for specific terms. This is discussed again in terms of a test problem in Section 11–4.

Timestep splitting works well in most situations even though there are a number of caveats and considerations that have to be made. This is the approach we have generally advocated throughout this book by describing different solution methods for different types of terms in the reactive-flow equations. Guidelines for implementing timestep splitting are given in Sections 11–2 through 11–4.

11–2. Coupling Physical Processes in Reactive Flows

This section and Sections 11–3 and 11–4 are a practical guide to coupling numerical representations of the physical and chemical processes in the reactive-flow equations by timestep splitting. Despite occasional reference to specific algorithms in the examples given, the information presented about coupling processes is quite general. A reactive-flow program should be an assembly of relatively independent software modules, each representing a particular physical or chemical process (or perhaps combination of processes) in the conservation equations. These modules should be as separate and self-contained as possible so that they may be reconstructed and changed independently by the users. This modular approach to building complex computer programs stresses flexibility and robustness in the algorithms and techniques. The specific modules are the enduring elements of the effort, rather than completely integrated simulation models. Algorithms for specific types of terms can also be updated and new physical processes can be added with minimal interference in the overall structure of the simulation program.

A consequence of this modular approach is that numerical errors peculiar to the solution procedure for one process can be isolated and not disastrously affect the solution of other processes. Consider a spark created by an electrical discharge that leads to flame ignition in a combustible gas. When a convection algorithm is used to compute the fluid density and other relevant quantities, any undershoots or overshoots in the newly computed density, momenta, or energy could result in local temperatures that are too low or too high. If the temperature is too high, there can be a runaway effect in which the material ignites too quickly. If the temperatures are too low, ignition may never occur. The general philosophy that has evolved to deal with the inevitable problem of local inaccuracies in the solution is to "protect" each part of a computation from the input it receives from the other process modules. Several specific implementations of this approach are given later.

We now describe an example of a simulation model that couples the generic processes for which individual solution algorithms were introduced in Chapter 4: convection with source terms, species reactions with subsequent energy change, and diffusion. The

discussion is then expanded to include thermal conduction, viscous diffusion, and molecular diffusion, so that the result is a fully compressible Navier-Stokes model. We then extend the discussion by describing how to construct a program with implicit fluid dynamics and multiphase flow.

11–2.1. Coupling for Fluid Dynamics, Species Reactions, and Diffusion

The simplest type of reactive-flow model describes compressible gas-phase flow with species reactions. Such a program solves the coupled, partial differential equations for convection and diffusion of momentum, energy, mass, and species densities (Chapters 7, 8, and 9) and a set of ordinary differential equations representing the species reactions (Chapter 5). Table 11.1 presents an outline for a modular reactive-flow model coupling these processes. The procedure first initializes the variables and then executes a series of distinct substeps within each timestep to (1) establish or modify the grid; (2) select the overall timestep; (3) compute the energy, mass, or other source terms; (4) evaluate convection; (5) integrate the species reactions; and (6) integrate the diffusion processes. Processes (2) through (4) could be designated as $\mathbf{G}_i$ $i = 1, \ldots, 4$ in equation (11–1.7). The partial results are then coupled, and the physical variables updated and checked for consistency.

The computer program combines a series of optimized modules representing the different physical processes, and then couples the results of these modules through timestep splitting. The particular formulation presented here is far from unique; variations are possible and sometimes preferable. The major choices that need to be made concern (1) timestep control, (2) the sequence in which the various physical-process substeps are performed, and (3) when and how the physical variables are updated as new information is generated by the separate substeps.

Substep 1. Establish the Appropriate Computational Grid

This is usually done at the beginning of a timestep and is based on knowledge of how the variables have been evolving from previous timesteps. In algorithms using an adaptive grid, the grid may be modified from step to step to allow clustering or refinement of computational cells at selected locations. Modifying the grid within evolving gradients or reaction zones usually introduces large errors, so predictive estimates in the computation should be used to determine where the resolution will be needed in subsequent steps. This is a problem-dependent process. Currently there is no set of all-purpose criteria that can be used to refine, coarsen, or move a grid.

There should not be too much change from one timestep to the next in adapting the grid. What constitutes "too much" is problem and method dependent. For example, if a sliding-grid clustering algorithm is used, it is important to prevent cell boundaries from crossing the location of adjacent cell boundaries during a single timestep. That means that a cell boundary must not move more than about one quarter of a cell per timestep so cell sizes cannot change by more than a factor of two between timesteps. In AMR methods that refine or coarsen computational cells by a factor of two in each direction, care has to be taken to make sure that the physical fluxes between adjacent cells remain essentially unchanged during the adaptation process. The key to this is how the fluxes and hence the variables are interpolated from the old to the new grid.

Table 11.1. A Global Timestep in a Reactive-Flow Program: Explicit Convection, Species Reactions, and Diffusion

Initialize variables.
∗ Start timestep loop
1. Establish the appropriate grid for the computation
2. Select Δt_g, where $t^n = t^o + \Delta t_g$ will be the end of the timestep
3. External energy deposition ($\mathbf{G}_1$)
   Problem dependent, for example,
   $T^{(0)} \to T^{(1)}$, $P^{(0)} \to P^{(1)}$, $\{n_l\}^{(0)} \to \{n_l\}^{(1)}$, etc.
   Variables now labeled with superscript (1)
4. Convective transport ($\mathbf{G}_2$)
   Convect variables (1) from t to $t + \Delta t_g$
   $\rho^{(1)} \to \rho^{(2)}$, $(\rho v)^{(1)} \to (\rho v)^{(2)}$, $E^{(1)} \to E^{(2)}$
   Compute $v^{(2)}$ from $(\rho v)^{(2)}/\rho^{(2)}$
   Compute internal energy $\epsilon^{(2)}$ from $E^{(2)}$, $(\rho v)^{(2)}$, and $\rho^{(2)}$
   Use equation of state to find $T^{(2)}$, $P^{(2)}$, and $\gamma^{(2)}$
   Variables now labeled with superscript (2)
5. Species-reaction kinetics (possibly subcycled with a shorter timestep) ($\mathbf{G}_3$)
   Evaluate the species reactions from t to $t + \Delta t_g$
   † $\{n_l^{(2)}\} \to \{n_l^{(3)}\}$, and $T^{(2)} \to T^{(3)}$ when solving equation for T
   † Use equation of state to find $\epsilon^{(3)}$, $T^{(3)}$, $P^{(3)}$, $\gamma^{(3)}$
   Variables now labeled with superscript (3)
6. Diffusion processes (molecular, thermal conduction, and viscosity) ($\mathbf{G}_4$)
   Evaluate the diffusive transport from t to $t + \Delta t_g$
   † $\{n_l^{(3)}\} \to \{n_l^{(4)}\}$, and $T^{(3)} \to T^{(4)}$ when solving equation for T
   † Use equation of state to find $\epsilon^{(4)}$, $T^{(4)}$, $P^{(4)}$, $\gamma^{(4)}$
   Variables now labeled with superscript (4)
7. Update the variables and prepare for new timestep
   $\rho^{(0)} = \rho^{(4)}$, $(\rho v)^{(0)} = (\rho v)^{(4)}$, $E^{(0)} = E^{(4)}$
   $T^{(0)} = T^{(4)}$, $P^{(0)} = P^{(4)}$, $\{n_l^{(0)}\} = \{n_l^{(4)}\}$, $\gamma^{(0)} = \gamma^{(4)}$

Start new timestep (go to ∗ above) with new variables labeled (0).

† If the species reactions must be subcycled, replace the two †'s with:
∗∗ Subcycle as needed –
Determine Δt_{ci}, for $\Delta t_{ci} < \Delta t_g$
Integrate the reaction equations over Δt_{ci}
Use the equation of state to find $\epsilon^{(3i)}$, $T^{(3i)}$, $\{n_l^{(3i)}\}$
Go to ∗∗ and continue loop until $\sum_i \Delta t_{ci} = \Delta t_g$
Use equation of state to find $\epsilon^{(3)}$, $T^{(3)}$, $P^{(3)}$, $\gamma^{(3)}$

Substep 2. Select the Global Timestep Δt_g

Table 11.1 shows that each physical process in this example, that is, external energy deposition, fluid convection, species reactions, and diffusion, is integrated for the *entire* time interval, Δt_g. Intermediate (provisionally updated) values for the physical variables can be determined after each process, using the equation of state as necessary. The question is how to set Δt_g as large as possible and still have sufficiently accurate solutions. If Δt_g is chosen as the smallest of the timesteps required to maintain stability and

accuracy of the different individual algorithms in the reactive-flow program, this timestep could be too small for the overall computation to be performed practically. This happens, for example, when Δt_g is chosen as the timestep of the species reactions and some of the species reactions are stiff. One approach to solving this problem is to use the Δt_g determined by the convection algorithm and to *subcycle* other processes requiring a smaller timestep. The global timestep Δt_g can be divided into a different number of substeps for each process requiring a shorter timestep. This is commonly done within the ODE solution for species reactions, even when no other reactive-flow process is present.

The global timestep Δt_g also cannot simply be taken as the largest of all the values required by stability for the distinct processes. If Δt_g is too large, there are serious problems with the accuracy, and the overall algorithm can be unstable even though none of the separate processes are. In practice, a first estimate of Δt_g is usually based on the CFL condition for the algorithm for the fluid flow, that is, $\Delta t_g = \Delta t_{cfl}$. Nonetheless, sometimes this gives a Δt_g that is too large. If there is substantial energy release or energy deposition somewhere in the computational domain, it may be necessary to reduce Δt_g. Another point to remember is that Δt_{cfl} is usually derived using rather ideal local conditions, such as a uniform grid and planar, one-dimensional flow. If there is strong curvature in a moving front in the flow, as occurs in converging shocks, it is sometimes necessary to reduce the timestep considerably (Oran and DeVore 1994). Section 11–2.2 extends this discussion of timestep control.

Substep 3. Computing External Sources of Mass, Momentum, and Energy

External source terms in the reactive-flow equations often completely drive the problem. Energy deposition may be a model of an ignition source such as a laser, spark, or hot wall. External forces, such as gravitational, electrostatic, and electromagnetic forces, usually appear as simple sources in the momentum equation. Extra mass sources could describe fluid injected through an orifice. Such processes may change the density, temperature, or species composition. Using external sources is important because it is a way of capturing the essence of a complex process and representing the effects of that process to the computed systems. In some cases, these are effectively implemented as boundary conditions. In its simpler forms, timestep splitting actually treats the results of some substeps as source terms for the others.

A source term can be important in determining the overall timestep. For example, if a large amount of energy is deposited locally or if energy is deposited very quickly, it may be necessary to reduce Δt_g. In Chapter 4, such local source terms were considered and used to introduce explicit and implicit integration techniques.

Substep 4. Computing the Convective Transport

Evaluating the effects of convection involves finding new values for the densities, momenta, and energies, and from these the velocities, pressures, and temperatures at each location on the computational domain. A major advantage of timestep splitting is the freedom to use an accurate monotone, positivity-preserving algorithm for convection. It is important to minimize numerical errors for calculations that need to run many thousands

or millions of timesteps. It is also important to maintain steep gradients when species reactions and energy release are coupled to convection.

Table 11.1 indicates a certain order in which the different physical processes are evaluated within the global timestep. Ideally, the order of evaluating the different processes should not matter. Invariance to the order of the substeps in timestep splitting is also a good test of accuracy and convergence. There may, however, be decided benefits to a particular order. For example, sometimes a particular physical process can be treated as a source term in a continuity-equation solver used for convection. Then it is necessary to evaluate appropriate coefficients and fluxes associated with this process before the convection. Examples of such effects are nonstiff species reactions and certain types of fluxes used as source terms in the Navier-Stokes equations. There are also advantages to evaluating convection at the beginning of the global timestep, as it is typically the most difficult of the processes to model accurately and, therefore, is often the error-determining process. When convection is computed first, the solution errors may then be immediately checked to see if it is necessary to reduce the timestep.

Errors in convection also can cause even larger errors in the other processes. In coupling energy release from the species kinetics to fluid dynamics, unphysical oscillations in the fluid variables from nonmonotone convection can cause major problems for kinetics integrations. Overshoots or undershoots in primary conserved quantities often lead to even larger errors in derived quantities, such as temperature. In addition, because the species reaction rates have an exponential form, unphysical overshoots can result in energy release much sooner than it should occur. This, in turn, feeds back into the convection, and the result can be runaway kinetic reactions and unphysical flame or detonation velocities. Therefore, convection in a simulation with energy release due to reaction kinetics must be more accurate than is usually required in nonreacting-fluid calculations.

Substep 5. Computing the Effects of Species Reactions

The species rate equations are then integrated through the interval Δt_g. This may require internal subcycling at some spatial locations where there are fast reactions and significant energy release. An example of subcycling the species reactions is described as a note at the bottom of Table 11.1. Subcycling in general is discussed in more detail in the next section.

Chapter 5 stressed the importance of using a method for solving ODEs that is self-starting and accurate. Information from prior species-integration steps cannot be required because the input to start the integration must come from the previously evaluated processes in the timestep-split procedure. For standard hydrocarbon combustion problems coupled to fluid dynamics, we have recommended a method such as CHEMEQ, discussed in Chapter 5. CHEMEQ is self-starting: it can be restarted at the beginning of each Δt_g with no appreciable start-up time and no information required from any but the most recently completed timestep. Since it uses only the most current values of the temperature and the species densities, there is no penalty in computer storage. It is also relatively inexpensive because no Jacobians need to be evaluated and there are no matrix inversions.

There are some problems for which CHEMEQ simply does not work well. This is the case when there are reactions in stiff equilibrium, when there is a very strong temperature dependence in the reaction rate, or when using reduced reaction mechanisms, which are

often stiffer than the original set of equations. In such cases, it may be necessary to adopt another method, such as YASS (also described in Chapter 5), which is more expensive but can treat such equilibria more effectively. The price for taking longer timesteps is having to set up and evaluate the Jacobian matrix at each timestep at each spatial location. Integrating each timestep then can take considerably longer than if no matrix operations are required.

Even using all of the possible devices, the kinetics integration may still cost a factor of four to forty more than the fluid dynamics, depending on the number of reactive species and the stiffness of the equations. To date, techniques to reduce the cost of integrating ODEs are hampered by large increases in required storage. Table-lookup routines, for example, could be used instead of continually evaluating reaction rates with costly exponentials. These tables, however, require extensive computer memory even for modest reaction-kinetics systems. Another approach that is very effective for large reactive-flow computations is to take advantage of what can be gained by parallel processing. Figure 11.1 is taken from a multidimensional computation that uses a detailed chemical-reaction mechanism for hydrogen-oxygen combustion. The implementation of an optimized method for load balancing among processors has brought the time of the integration of a full hydrogen-oxygen reaction mechanism down to less than required to integrate the fluid dynamics.

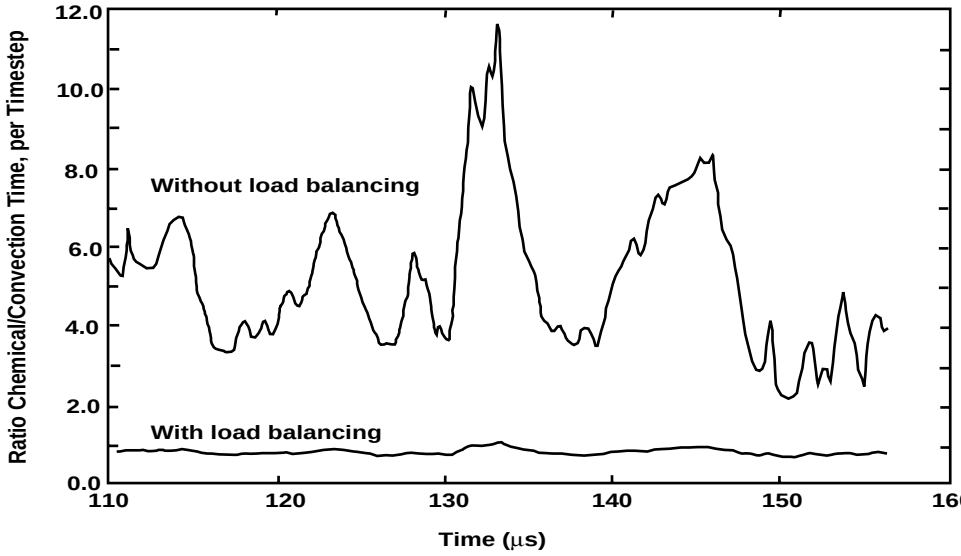

Figure 11.1. An optimized, load-balanced program implemented on a parallel-processing computer can significantly reduce the computational time required to integrate even substantial chemical reaction sets. The figure shows two profiles of the ratio of the computer time required for integrating the chemical reaction rates, τ_{chem}, to the time required to integrate the fluid dynamics, τ_{fd}, as a function of timestep. The simulation described a multidimensional detonation propagating in a mixture of hydrogen, oxygen, and argon (Weber et al. 1997). The upper curve is the ratio that would normally be expected for this integration of a stiff chemical reaction mechanism with nine chemical species and fifty reaction rates. The lower curve, the same computation but with an optimal load-balancing among the processors, shows how τ_{chem} has been reduced to even less than τ_{conv}. The fluid-dynamics integration includes the time advancement of density, momenta, and energy, plus convection of the nine chemical species.

Achieving such results requires a very large computation, one that will justify the use of a massively parallel computer, and a program that is scalable on such a computer. Even then, stiff reaction kinetics will generally require more time than the rest of the computation.

At the end of this substep, the released heat is converted to an effective pressure change at constant volume because the cell volume has been held fixed during the kinetics calculation. The energy change and the temperature and pressure this implies can be calculated from the heats of formation and the changes in the number densities of the various interacting species, as described in Chapter 5. Except for the $\{n_l\}$, the variables are not updated at this stage. The temperature is held constant during the species-reaction substep, and changed only at the end of the substep.

Substep 6. Computing the Effects of Diffusion

Diffusion is generally a smoothing process. Nevertheless, the simplest explicit implementations have stability limits. The effects of molecular diffusion may be relatively fast, especially when there are light, fast-moving species present, such as hydrogen atoms or molecules. In such cases, we can generally achieve efficient solutions by subcycling the molecular-diffusion process. Otherwise, the presence of fast diffusion terms in the equations requires Δt_g to be reduced below what would normally be used for kinetics-convection problems alone. Exactly how many subcycles are required can be determined by estimating the relative timesteps of the various contributing processes. There are also cases in which subcycling becomes expensive and it is more economical to use an implicit method for the conduction terms.

Using an explicit time-integration algorithm generally restricts the timestep to a fraction of the acoustic transit time across the smallest cell. In many reactive flows, this means that physical diffusion and thermal conduction can then be integrated accurately using a simple explicit algorithm. There are, however, situations in which the sound speed is not the fastest characteristic velocity in the system, and yet the integration is performed on the acoustic timescale. This situation occurs in certain plasma systems, for which the temperature is high and the atoms are either partially or fully stripped of their electrons. When the thermal conduction is dominated by the very fast motions of the free electrons, it must be treated implicitly. In addition, electron thermal conduction varies as $T^{5/2}$ and thus is even more nonlinear than the $T^{3/2}$ dependence considered in Chapter 7. Algorithms for nonlinear thermal conduction and for multidimensional diffusion are described in Sections 7–3 and 7–4.

Sometimes the best approach is a combination of explicit and implicit conduction. Consider the problem in which there is an intense laser interaction with a material, such that the temperature may vary by tens of electron volts over a distance of a few microns (see, for example, Gardner, Bodner, and Dahlburg [1991]). Then the heat-conduction equation could be integrated implicitly in the direction of the incident laser, where the gradients are steepest and the cell sizes the smallest, and explicitly in the other direction.

Substep 7. Updating Variables

At the end of the timestep Δt_g, all variables are updated appropriately, making sure that the densities, energies, pressures, and temperatures are consistent. Section 11–2.3 below

describes conservation checks and renormalization techniques used to ensure that synchronization errors in the convection and inconsistencies in asymptotic integration formulas for reaction kinetics will not adversely affect the overall integration results. This is also the point at which diagnostic outputs should be computed since all of the quantities have been assembled at the same time and the same stage of integration.

11–2.2. Timestep Control

The first step in selecting a global timestep Δt_g is to estimate the individual timesteps, $\{\Delta t_m\}$, required by the various algorithms used the solve for the contributing processes m, where $m = 1 \ldots M$. As discussed in Chapter 4, the choice is constrained by both the physical scales of the problem and the choice of numerical algorithm. Consider, for example, the program described in Table 11.1. If each type of term is evaluated by an explicit algorithm, we define

$$\Delta t_g = \min(\Delta t_{conv}, \Delta t_{visc}, \Delta t_{cond}, \Delta t_{md}). \tag{11–2.1}$$

Then using explicit algorithms for these processes,

$$\Delta t_{conv} = C_{conv} \min\left(\frac{\Delta}{|v| + c_s}\right) \tag{11–2.2}$$

$$\Delta t_{cond} = C_{cond} \min\left(\frac{\Delta^2}{2\lambda/\rho c_p}\right) \tag{11–2.3}$$

$$\Delta t_{visc} = C_{visc} \min\left(\frac{\Delta^2}{2\mu/\rho}\right) \tag{11–2.4}$$

$$\Delta t_{md} = C_{md} \min\left(\frac{\Delta^2}{2D}\right). \tag{11–2.5}$$

Here Δ is the computational cell size and the subscripts *conv*, *visc*, *cond*, and *md* stand for convection, viscosity, conduction and molecular diffusion, respectively. The coefficient C_{conv} is the CFL number for the particular convection algorithm, and the other C_m's represent the equivalent for the algorithms used for those processes. Note that the minimum taken in each of the equations (11–2.2) through (11–2.5) represents a minimum evaluated over each computational cell in the problem. (The variables and coefficients were defined in Chapter 2, Table 2.1.)

If a particular process m is solved implicitly, it might be possible to remove Δt_m from consideration in equation (11–2.1). For example, if convection is solved by an implicit algorithm, $\{\Delta t_g\}$ would then be determined by diffusion processes. If all of the terms are solved implicitly, $\{\Delta t_g\}$ is determined by the sizes and scales of the phenomena that have to be resolved.

Even though there are ways of estimating the appropriate timestep for each physical process individually, usually based on the stability limit of the algorithm used, there is no definitive way to determine the appropriate timestep for the whole system when several processes are interacting. Nonetheless, the nature of the interactions between

processes often allows the overall timestep to be larger than the minimum required by one of the contributing processes. If stability of the solution algorithm is an issue for one of the submodels, that submodel might be solved implicitly or subcycled, as discussed in the next subsections. This is the case in many reactive-flow programs, where the stiff reaction rates often require a timestep much shorter than the convection timestep. We can then use the CFL condition to determine Δt_g and then subcycle the reaction kinetics within this timestep.

Timestep Control for Particular Physical Situations

An interesting situation would be the inverse, where the flow velocities are very slow, the species reactions are slow, and where the sound waves that must be resolved are relatively fast. In this case, it might be useful to reverse the process, and let the species reactions determine Δt_g, and subcycle the fluid dynamics. Other cases can be imagined in which the overall timestep should be smaller than estimates based on individual processes. Physical systems are usually in a state where two or more processes compete and nearly cancel. Nonlinear coupling between processes can require a smaller timestep, as when two stiff reactions compete on about the same time scale. Usually, reducing the timestep by a factor of two solves this problem.

For strong shocks, or when a strong shock couples to energy release, it is often necessary to reduce the timestep. One useful correction to the timestep is, at each new timestep, to multiply the sound speed used to evaluate the timestep by a factor that accounts for the shock jump condition,

$$N_{cfl} = c_s \frac{P_{\max}}{P} \frac{\Delta t}{\Delta x} \tag{11–2.6}$$

where N_{cfl} is the CFL number (Williams 1997). Another approach useful for simulations of detonation is to define another criterion to be used in conjunction with the CFL condition, so that there are two numbers (Gamezo and Oran 1997),

$$N_{cfl} = \frac{\Delta t}{\Delta x} c_s^{\max} \quad \text{and} \quad N_{rr} = \frac{\Delta x}{c_s^{\max}} W^{\max}, \tag{11–2.7}$$

where $W^{\max}$ is the maximum reaction rate so $1/W^{\max}$ is a characteristic reaction time. A reasonable criterion seems to be to have $N_{cfl} < 0.5$ and $N_{rr} < 0.2$, in which case the energy released goes into at least five cells in ten or more timesteps.

Subcycling the Integration of Selected Processes

An important technique for improving accuracy and stability without using the most restrictive Δt_g for all of the processes is to subcycle the process integrations within Δt_g. This is useful when the changes from some of the processes are slow enough to use Δt_g, but a shorter timestep is required for stability or accuracy of other processes. For example, consider a series of processes, each characterized by a Δt_i, each corresponding to one of the $\mathbf{G}_i$'s in equation (11–1.7). The Δt_i for $\mathbf{G}_i$ is determined by the particular physical process in the equation and by the stability and accuracy criteria of the method used for that particular substep. When $\Delta t_i \ll \Delta t_g$, the $\mathbf{G}_i$ integration is subcycled until Δt_g is reached. If $\Delta t_i > \Delta t_g$, then Δt_i is reduced to Δt_g. This timestep-splitting and subcycling procedure

assumes that $\mathbf{G}_i$ can be decoupled from the other processes for the duration of Δt_g. The adequacy of this assumption may be tested by running the program with two different global timesteps and comparing the answers.

Consider several examples. For the species integration, Δt_c is determined by the particular ODE integrator. Accuracy within Δt_c is controlled by predetermined convergence parameters that change the stiffness criterion, the timestep, and thus effectively the degree of conservation. It is important that not too much change occurs, either in the particular species densities or in any energy changes they imply, during Δt_g. Another example is the diffusion of light, fast-moving hydrogen atoms in a relatively slow-moving flow. Subcycling is essential for the molecular diffusion of the hydrogen, but may not be necessary for the thermal conduction, whose effects are derived from averages over all of the species present. Exactly how often it is necessary to subcycle here depends on the grid and the timestep. Again, it is important that the overall changes in the physical variables during Δt_g are not too large.

In each of the subcycles associated with a species integration, the temperature can vary. Thus, in addition to solving the species equations, it is necessary to solve either an algebraic equation or an ODE for the temperature. Section 5–4.1 described a method that could be used when the enthalpies of the species are known as a function of temperature. Then integrating an additional, usually stiff ODE for the change in temperature can be avoided. This is done by assuming that the total energy E is conserved, so that the internal energy ϵ can be written

$$d\epsilon = \sum_l \left(n_l^{\text{new}} - n_l^{\text{old}} \right) \Delta h_{o,i} \tag{11–2.8}$$

$$\epsilon_{\text{new}} = \epsilon_{\text{old}} - d\epsilon. \tag{11–2.9}$$

Then ϵ_{new} is used with the $\{n_l\}$ in the equation of state in $\epsilon_{\text{new}} = \mathcal{F}_1(T, \{n_l\})$ to find T.

11–3. More on Coupling: Warnings and Extensions

Each algorithm used to integrate a particular $\mathbf{G}_i$ has its own peculiarities and uncertainties. There are also many ways of actually implementing an algorithm. The general philosophy that has evolved in applying timestep splitting is to "protect" each type of interaction from the errors that might have occurred in previous substeps. Sometimes this means simply imposing a minimum or maximum on the temperature and pressure passed from one substep to another. Sometimes the order in which the processes are updated becomes important.

Not every problem is a numerical error. For example, sometimes when an algorithm does not converge, it is a signal reflecting a basic error in the physical assumptions. This may only become evident during the computational process. An example of this is when the algorithm does not produce a physically reasonable, converged steady-state solution to a fluid-dynamics problem: the problem itself could be inherently unsteady. For this reason, it is important to understand the sources of numerical problems so that they can be dealt with and eliminated (or at least avoided). Here we describe some of these possible

problems, suggest solutions, and then proceed to describe direct extensions to include multidimensional effects and other physical processes.

11–3.1. General Comments

Ways of Incorporating Diffusive Fluxes

There is an interesting choice of how to incorporate the effects of thermal conduction, viscosity, and molecular diffusion. First, it is possible to evaluate these diffusion terms individually, and then to use the results to update appropriate variables. In this case, each term is a different substep in Δt_g. This involves separately computing the fluxes for each type of diffusion process and then using these fluxes in a difference approximation.

Another approach is to evaluate the fluxes for the different processes, save them, and then use them as source terms in the convective-transport equations in Substep 6. This does not save much work, and it could put unwanted restrictions on the fluid-dynamics substep. The motivation is to help keep the solution monotone and avoid conflicts that might arise in synchronizing the results of the different processes. To date, in the limited number and types of tests we have done on combustion problems, there does not appear to be much difference, either in the answer or the amount of work done.

Multidimensional Direction-Split Convection

As described in Chapter 8, the principles of splitting can be used to produce algorithms for multidimensional flow when it is impractical or too expensive to use the fully multidimensional algorithms. The convective derivatives in different directions can be treated as separate processes, updating the flow variables very much as they were updated for timestep splitting in Table 11.1. This has often been called *direction splitting*. A two-dimensional or three-dimensional program can be constructed by using a one-dimensional algorithm in each direction. Three-dimensional models can also be constructed by combining a two-dimensional and one-dimensional integration. In these cases, the equations for the direction being integrated have source terms representing the changes due to the directions not being integrated. Table 11.2 is an outline of the steps and coupling involved in a multidimensional timestep-split program that uses direction splitting and includes the effects of gravity.

Direction splitting produces some directional asymmetry in the computations. This unwanted asymmetry can be reduced by alternating the directions of integration within subsequent timesteps. For example, during one Δt_g, integrate first in the x_1-direction, and then in the x_2-direction. Then in the next Δt_g, integrate first in the x_2-direction, and then in the x_1-direction. Even more of the asymmetry can be eliminated by doing two computations within a timestep and averaging the results. That would mean integrating in the x_1-direction followed by the x_2-direction, then using the starting values and reversing the process, all within the same timestep. If the problem being solved is that sensitive, it is usually worthwhile to avoid direction splitting and use a fully multidimensional algorithm instead of essentially computing every timestep twice. Very often, just alternating directions during each global timestep is sufficient.

Table 11.2. Multidimensions and Gravity

Initialize variables.
∗ Start time loop.
1. Determine the appropriate grid for the computation
2. Determine Δt_g
   Variables now labeled with superscript (0)
3. Convective transport
   (Definition of x_1, x_2, and x_3 depends on Δt_g)
   a. Convect variables (0) in x_1-direction
   Variables now labeled with superscript (1)
   b. Convect variables (1) in x_2-direction
   Variables now labeled with superscript (2)
   c. Convect variables (2) in x_3-direction
   Variables now labeled with superscript (3)
4. Gravity: as described in text
   Variables now labeled with superscript (4)
5. Another process and updating . . . .
6. . . .
7. . . .
8. . . .
9. Update variables

Start new timestep (go to * above).

Inconsistencies in Convection and Reaction Algorithms

After the convective-transport substeps are computed using a monotone algorithm, there is often an inconsistency among the computed densities. For a fluid density ρ_i at location i,

$$\rho_i \neq \sum_{l}^{N} \rho_{l,i}, \tag{11–3.1}$$

where the summation is over all species l. The nonlinearity in the flux limiter used in the solution of each continuity equation allows small variations, each of which depends on the shape of the different species profiles. This inequality, or "synchronization" error, can occur even though the convection algorithm is conservative over the entire grid, that is, even though ρ_l and ρ are separately conserved, and the global conservation laws are individually satisfied.

When this local disparity between the total density and the sum of the species densities is important, the species densities can be renormalized to be consistent with the total mass density. A commonly used renormalization is

$$n_l^{\text{new}} = n_l^{\text{old}} \left(\frac{\rho}{\sum_m \rho_m} \right) \tag{11–3.2}$$

for $l, m = 1, \ldots N$ for each computational cell. This renormalization is locally and globally conservative. Other renormalizations have been suggested and tried, but this particular one is usually used. Since most renormalizations are empirical, care must be taken in

their use. In some cases, using equation (11–3.2) leads to significant errors in computing minor species' densities. This could be the case, for example, in the computation of NOx formation in hydrocarbon combustion. Some of the particular difficulties in computing multispecies flow at sharp gradients have been discussed by Ton (1996).

Conservation in Species-Reaction Integration

Most methods that integrate stiff ODEs are not perfectly conservative. For example, consider a chemical-reaction integration, and count the number of atoms in the system before and after integration. Any unphysical discrepancy indicates the accuracy of the integration method. When this lack of conservation must be controlled, the usual technique is to tighten the convergence criteria. Unlike the synchronization errors that appear locally to be incorrect but that globally conserve the integrated species masses, these errors usually do not ensure global conservation.

Furthermore, any formulation designed to treat the different equations differently is likely to violate conservation. In some integration methods, different linear formulas are used for different equations. If any one of the formulas were used for the entire system, the solution would be conservative. The combination, however, is not. In manifold-integration techniques (Chapter 5), conservation is usually not guaranteed rigorously. Also, algorithms such as CHEMEQ that are based on asymptotic formulas to obtain long timesteps often violate conservation to some extent. The general procedure for renormalization given earlier is generally accurate enough to fix the problem.

Equation-of-State Computations

As discussed in Chapter 5, the way that the new temperatures and pressures are found in combustion problems is to assume that the internal energy ϵ is constant, the species $\{n_l\}$ are known, and an iteration can be performed on $\epsilon = \mathcal{F}_1(T, \{n_l\})$ to find T. The function $\mathcal{F}_1$ can be derived from the expression for the specific heats or enthalpies, which are generally fit to a polynomial in T. The pressures are found from the equation of state $P = \mathcal{F}_2(\{\rho_i\}, T)$, where P could be an ideal-gas equation of state. Then, for example, γ may be evaluated from

$$\gamma = \frac{c_p}{c_v} = \frac{\sum_l n_l \dfrac{dh_l}{dT}}{\sum_l n_l \left(\dfrac{dh_l}{dT} - k\right)}. \tag{11–3.3}$$

When the system is in thermodynamic equilibrium, a quantity such as energy or enthalpy is conserved, and have a closed-form solution for the equation of state. There is an algebraic equation to solve rather than a differential equation.

While this prescription, or something similar based on other kinds of imposed conservation conditions, works well for most combustion problems, it is not universal. For many reactive-flow problems, evaluating the temperature or densities can be a considerably more complex operation. Examples of such systems include any system characterized by multiple temperatures, as is the case with various regions of the earth's atmosphere, the atomic reactions in the solar corona, or many plasma-dynamics problems. In such situations, it may not be reasonable to assume that there is thermal equilibrium among the electrons, neutral species, or ions, or even the various rotational and vibrational states of

the constituents. Sometimes it is possible to assume a Maxwellian distribution represents the temperature of each separate type of reacting species, say electrons and ions, and that there is some exchange of energy among these. The result is that, in addition to the set of species-interaction equations, one or more additional ODEs are required to represent changes in temperatures. These extra equations usually include an energy-transfer term because of collision among different types of particles.

The general prescription given in Table 11.1 works even if the equation of state is quite complex and requires considerably more work to evaluate than is required for an ideal gas. Condensed-phase materials, for which the equations of state are extremely complex, require iteration, and are often somewhat empirical. Thermonuclear materials have a deceptively simple equation of state that is very sensitive to small changes in the quantities. These equations of state may have complex forms that are often presented in the form of tables, which must be interpolated in density and internal energy to find temperature and pressure. In situations in which complex equations of state must be used, it is important to insure that the numerical representations are continuous and appropriately single valued. Surprising and "mysterious" numerical problems occur where the transitions in the form of the equation of state are not smooth. In these cases, too, the general framework is the same as described in Table 11.1, but now with specific calls to equation-of-state routines.

Coupling Convection and Species Reactions

The accuracy of fluid variables in reactive regions must be monitored even when a high-order monotone method is used. It is generally not a good idea to disrupt the fluid calculation by permanently modifying extreme values of energy, momentum, or density that might occur during a timestep. It is better to limit the derived temperature that is actually used by the species integrator. One way to do this is to pass the species-integration routines a temperature that is a weighted average or a minimum of the locally calculated temperature at each computational cell and the surrounding temperature values.

For example, for most reactive-shock computations, we have been able to use

$$T_{\text{new},i} = \min(T_{i-1},\ T_i,\ T_{i+1}). \tag{11–3.4}$$

Occasionally it is necessary to extend the range of temperatures even further to $T_{i\pm2}$. Equation (13–3.4) is extended to multidimensions by including minima in a region in all directions. This "fix" is usually adequate to restrict the temperature from above. It is also useful in both species and convection integrations to limit the temperature from below and above. For example, consider using $T_{\min} = 250$ K and $T_{\max} = 3{,}000$ K. Another approach is to convect the fluid entropy as well as energy with the species concentrations. In this way the temperature in each cell can be prevented from dropping below an adiabatically compressed value that are determined from the local density and entropy.

In contrast to the accuracy criteria for convection, which might have to be tightened when there is significant energy release, convergence requirements for the reaction integrators can often be *relaxed* somewhat in reactive flows. A convergence criterion for the ODEs of $\epsilon = 0.001$ (Chapter 5) gives acceptable results when the equations are integrated alone. Nonetheless, when they are coupled to convection, which allows expansion and

rarefaction in the densities, temperatures, and pressures, the answers are acceptable with $\epsilon = 0.01$, and conservation may actually be improved. This has been tested to some extent for both flame and detonation calculations. Some of the advantages gained from reducing ϵ come from the nature of integrators, such as CHEMEQ, that work best when the system is slightly out of equilibrium. Fluid dynamics, with rarefactions, compressions, and fluctuations, tends to guarantee at least small deviations from equilibrium.

When a reaction wave – perhaps a detonation – passes through a region, there can be a substantial amount of energy released in a computational cell. If the grid is coarse and the timestep is chosen in accordance with the CFL condition, there can be more energy released at one time than the program can handle accurately. One way to treat this, when the timestep cannot be reduced further, is to limit the amount of energy released in a computational cell in any timestep. Sometimes the allowed fraction of energy release in a timestep is determined empirically, sometimes it is set as a fraction of the time at which sound crosses a cell. Recommended values of this fraction are approximately 0.1.

How to handle large amounts of energy release in a timestep is a very interesting problem in computational physics: we are trying to avoid situations in which the finite size of the computational cells leads to information transmitted unphysically at speeds that are much faster than the speed of sound in the material. A fundamental way to approach this problem would be to use kinetic theory to find reasonable limits on the cell-integrated reaction rate for these situations. For the present and for simplicity, however, we use this physically motivated "patch."

11–3.2. Including Gravity

There are two different ways to include the effects of gravity. The first uses the idea of a *gravitational potential energy*. Then the energy conservation equation contains an expanded definition of the energy density,

$$E^* \equiv \frac{1}{2}\rho v^2 + \epsilon + \phi(\mathbf{x}, t). \tag{11–3.5}$$

For example, the gravitational potential can be written as $\phi = \rho g h$ in a planar geometry with constant gravity. More generally, the gravitational potential is defined as

$$\phi(\mathbf{x}, t) \equiv -\rho(\mathbf{x}, t) \int_{\mathbf{x}_o}^{\mathbf{x}} \mathbf{g}(\mathbf{x}) \cdot d\mathbf{x}, \tag{11–3.6}$$

where $\mathbf{g}(\mathbf{x})$ is the vector gravitational field and $\mathbf{x}_o$ is a reference location where the gravitational potential is taken to be zero.

The energy-conservation equation is written just as before, but uses the new definition of the energy density, E^*,

$$\frac{\partial E^*}{\partial t} = -\nabla \cdot E^* \mathbf{v} - \nabla \cdot P\mathbf{v}. \tag{11–3.7}$$

The form of the equation has not changed, but the definition of the total conserved energy has. There is also a vector force density, $\rho\,\mathbf{g}$, which must be added to the momentum equation.

In this approach, errors can accumulate from taking differences of large numbers. Further, when the physical system spans many gravitational scale heights, the gravitational potential term in the energy equation can exceed the internal and kinetic energies by large factors. It then becomes difficult to find the temperature accurately.

A second way to incorporate gravity is to integrate the equation for the internal plus kinetic energy without the potential-energy term included in equation (11–3.5). Many gravitational scale heights can be represented because the calculations are entirely local. In this approach, the momentum source term and the kinetic-energy source term from the gravitational acceleration must be computed consistently. One way to do this is to timestep split the gravitational terms in the momentum and energy equations and treat them together consistently. Let ρ^o, v^o, and E^o be the mass density, velocity, and energy density after one timestep calculated without gravity. The updated values of these quantities due to gravitational effects, ρ^n, v^n, and E^n, can then be determined from

$$\rho^n = \rho^o \,, \tag{11–3.8}$$

$$\rho^o(\mathbf{v}^n - \mathbf{v}^o) = \rho^o g \Delta t, \tag{11–3.9}$$

$$E^n - E^o = \frac{1}{2}\rho^o(\mathbf{v}^n \cdot \mathbf{v}^n - \mathbf{v}^o \cdot \mathbf{v}^o). \tag{11–3.10}$$

Here the kinetic-energy change due to the gravitational acceleration has been added. The temperature and pressure are not directly affected by gravity; the changes occur indirectly on succeeding timesteps.

11–3.3. Implicit, Lagrangian, and Navier-Stokes Fluid Dynamics

Implicit methods for convection are more complex and generally more difficult to implement than explicit methods. They can be very valuable because they can be used to remove the limitation on the timestep imposed by the acoustic transit time. Lagrangian methods try to reduce or eliminate numerical diffusion by moving the grid along with the fluid. They are fairly straightforward to implement in one dimension, but become problematic in multidimensions. Navier-Stokes solvers include viscosity and often other diffusive transport terms in order to treat boundary layers and other fluid-structure interactions. Each of these extensions to the reactive-flow Euler equations extends the range of problems the model can treat, and each has an additional cost or numerical problems associated with it. Here we consider some of the ways that the coupling algorithms for timestep splitting may have to be changed to accommodate implicit convection.

When the fluid velocities are relatively high, explicit, compressible fluid-dynamics methods are appropriate. In such cases, we can afford and might need to resolve the sound waves in the system. The explicit methods described in Chapter 8 and 9 apply. When we do not want or cannot afford to integrate the convection on the acoustic timescale, timestep-splitting procedures may have to be changed. One example of this is a simulation of a laminar flame, in which the fluid and flame velocities are generally a factor of several hundred slower than the sound speed. Another example is a shock or detonation system in which boundary layers become important and require fine spatial resolution. Resolving these boundary layers on the acoustic timescale may be prohibitive because the cells are so small.

Because of the stability of implicit algorithms for large timesteps, the approach described earlier for coupling Eulerian fluid dynamics can be made implicit in time. This would then mean that the convection substep (see, for example, Table 11.1) is replaced with a substep built around the prescription for using implicit fluid dynamics combined with an explicit convection algorithm such as FCT. There are some minor differences, but the general concept of updating all variables after each substep holds. This means that it can be relatively straightforward to interchange explicit and implicit algorithms, the major (and very significant) additional requirement being a good solver for elliptic equations. With implicit fluid dynamics, however, Δt_g can now be much longer than the acoustic timestep. This often makes an explicit integration algorithm for other processes unstable, thus requiring subcycling or other implicit solution procedures.

When the convection algorithm fluid dynamics are implicit, another coupling problem can result: the need to couple two separately stiff processes, both of which are solved implicitly. The physical situation that becomes difficult to solve is one in which two simultaneous processes can both move mass or energy much faster than one cell per timestep. For example, suppose one process moves energy rapidly into a region of space, and another process moves it away equally rapidly. The stiff dynamic equilibrium that results may be beyond treatment by the coupling method shown in Table 11.1. Methods for extending the timestep-splitting concepts to systems with multiple stiff processes are discussed in Section 11–4.

There are differences in coupling approaches for Lagrangian methods, such as those that would couple the ADINC algorithm (Chapter 9), in a full reactive-flow model. Table 11.3 shows how we have used a one-dimensional Lagrangian fluid-dynamics computation to create a program for calculating flame propagation. The timestep splitting used here appears similar to that described earlier, but it is quite different in a fundamental way. In this Lagrangian flow situation, both the reaction kinetics and sound waves are stiff. After each substep in Δt_g, the change of pressure or internal energy attributable to that process is accumulated. These provisional pressure changes are now added to the old pressure before the convection stage to obtain a new pressure to use in the fluid-dynamics algorithm. The entire provisional pressure change is introduced directly into the implicit convection substep to allow the fast acoustic time scale to redistribute the pressure appropriately.

This approach works because it allows fluid-dynamic expansion and, in the Lagrangian model of Table 11.3, diffusive transport to relax the pressure from the flame region on time scales characteristic of the sound speed. Thus the pressure stays effectively constant if the timestep is larger enough than the sound-transit time across a cell. In Lagrangian systems, the convection may still be effectively explicit because of the changing grid. Coupling may become more difficult when the more complex gridding is used with Lagrangian models that need to be run for long times. General unstructured gridding, often used to treat the large grid distortions needed to follow complex and turbulent flows in a Lagrangian manner, makes implementation of implicit methods on that grid very difficult and expensive. This is an ongoing research problem.

When the fluid-dynamic components of the reactive-flow model are extended to the full Navier-Stokes equations, the worst problem for coupling is the need to solve implicit diffusion. This is generally only necessary when very fine resolution is needed to resolve a boundary layer. In these cases, however, timestep splitting again leads to computational systems with two or more stiff processes.

Table 11.3. Coupling with Lagrangian Fluid Dynamics

Initialize variables.
∗ Start time loop.
1. Determine the appropriate grid for the computation
2. Determine Δt_g
   Consider all physical processes involved
3. External energy deposition
   All variables now labeled with superscript 1.
   Calculate change in internal energy $\Delta\epsilon_1$ and pressure ΔP_1
4. Chemical reactions
   Integrate from t to $t + \Delta t_g$
   Update $\{n_i(x)\}$
   Calculate $\Delta\epsilon_2$ and ΔP_2
5. Evaluate diffusive transport coefficients on the grid
   $\lambda_m(\mathbf{x})$, $\{D_{ij}(\mathbf{x})\}$, $D_i^T(\mathbf{x})$
6. Thermal conduction
   Integrate from t to $t + \Delta t_g$
   Calculate $\Delta\epsilon_3$ and ΔP_3
   Do not update any variables
7. Ordinary diffusion and thermal diffusion
   Integrate from t to $t + \Delta t_g$
   Only update $\{n_i(x)\}$
   Calculate $\Delta\epsilon_4$ and ΔP_4
8. Radiation transport
   Compute q_r and update energies, other variables
9. Convective transport
   Calculate the new pressure or internal energy,
   $P^n = P^o + \sum_i \Delta P_i$, $\epsilon^n = P^o + \sum_i \Delta\epsilon_i$
   Integrate from t to $t + \Delta t_g$
   Update all variables: ρ, P, T, E, $\{n_i\}$, γ.

Start new timestep (go to * above).

There are other numerical coupling problems associated with solving Navier-Stokes equations in timestep-split and direction-split representations. For Navier-Stokes systems, viscosity is the diffusive process that complicates the coupling. The diffusion associated with viscosity, that is, the ∇^2 terms, can be handled in direction-split models, but the cross terms such as $\partial V_x/\partial x\partial y$ require additional storage for even explicit implementations. (This can be done efficiently, but additional memory is required to pass the cross-derivative source terms across the domain boundaries for direction-split parallel processing.)

11–3.4. Multiphase Flows

Equations of multiphase flows were introduced briefly in Chapter 2. There are two general approaches for treating multiphase flows when the separate phases have different velocity fields: the continuum approach (Eulerian), in which the different phases are represented

by separate density and velocity fields, and the discrete, macroparticle approach (Lagrangian), in which one or more types of macroparticles are embedded in a flow. The momentum equations for each phase contain terms that describe the effect of one phase on the other, and these are variously called "coupling," "drag," or "friction" terms. In the macroparticle approach, each computational particle in some way represents a very large number of real particles. The simplest case, in which there is a single velocity field, is very similar to multispecies reactive flow and will not be discussed further here. The other case presents new and difficult coupling problems, some of which are briefly discussed here.

Consider a particle-laden gas flow with particles of two distinct sizes. The smaller particles, designated a, move at about the local fluid velocity because the drag forces are relatively large. The heavier particles, designated b, are sufficiently large that they can move differently from the fluid for a while after the fluid changes direction. We assume that the overall fraction of the volume occupied by particles is only a small percentage of the gas volume. Thus particle collisions are relatively rare. The physical coupling between the three phases involves an exchange of momentum and energy among the phases. We assume that the particle sizes are constant in time, so we can ignore the complications of evaporation and condensation that require mass transfer among phases. Momentum and heat transfer among phases are conservative, local processes in the sense that they do not involve spatial derivatives. Faster-moving particles slow down while accelerating the background gas. Hotter particles cool while heating up the background gas.

If there are enough particles, or we are concentrating on sufficiently macroscopic regions, it is usually adequate to treat each phase as a distinct fluid. Each of the particle "fluids" contributes essentially zero pressure to the pressure gradient, but each affects the overall pressure field through the drag terms. Assuming equal temperatures for the three components, the coupling in the momentum equations can be written as,

$$\frac{\partial \rho_g \mathbf{v}_g}{\partial t} + \nabla \cdot \rho_g \mathbf{v}_g \mathbf{v}_g = -\nabla P + C_{ga}(\mathbf{v}_a - \mathbf{v}_g) + C_{gb}(\mathbf{v}_b - \mathbf{v}_g) + \cdots \tag{11–3.11a}$$

$$\frac{\partial \rho_a \mathbf{v}_a}{\partial t} + \nabla \cdot \rho_a \mathbf{v}_a \mathbf{v}_a = C_{ga}(\mathbf{v}_g - \mathbf{v}_a) + C_{ab}(\mathbf{v}_b - \mathbf{v}_a) + \cdots \tag{11–3.11b}$$

$$\frac{\partial \rho_b \mathbf{v}_b}{\partial t} + \nabla \cdot \rho_b \mathbf{v}_b \mathbf{v}_b = C_{gb}(\mathbf{v}_g - \mathbf{v}_b) + C_{ab}(\mathbf{v}_a - \mathbf{v}_b) + \cdots . \tag{11–3.11c}$$

Although the force terms on the right side of these equations might also include other effects such as viscous forces and gravity, here we are focusing on the momentum-coupling terms, where C_{ga}, C_{gb}, and C_{ab} are the drag or coupling terms that depend on the particular physical properties of the phases. These terms must sum to zero to guarantee momentum conservation.

There are corresponding terms in energy equation that ensure conservation of energy. For example, written in terms of the change of kinetic energy,

$$\mathcal{E}_{ke} \equiv \frac{1}{2}\rho_a \mathbf{v}_a \cdot \mathbf{v}_a + \frac{1}{2}\rho_b \mathbf{v}_b \cdot \mathbf{v}_b + \frac{1}{2}\rho_c \mathbf{v}_c \cdot \mathbf{v}_c, \tag{11–3.12}$$

these appear as

$$\frac{\partial \mathcal{E}_{ke}}{\partial t} = -C_{ga}(\mathbf{v}_a - \mathbf{v}_b)^2 - C_{gb}(\mathbf{v}_g - \mathbf{v}_b)^2 - C_{ab}(\mathbf{v}_a - \mathbf{v}_b)^2 + \cdots . \tag{11–3.13}$$

The overall change in kinetic energy due to the presence of the momentum coupling is always negative. Then to conserve energy, there must be a corresponding heat generation and corresponding temperature rise. The viscous drag on the particles leads directly to heating, part of it going into internal energy and part of it into PdV work on the fluid.

The main problem encountered in the continuum multifluid representation occurs when any or all of the C_{ga} or C_{gb} coupling coefficients becomes large. This means that one or both of the particle species relaxes to the local fluid velocity very rapidly. As this relaxation gets faster and the numerical integration becomes more expensive, the value of having a separate velocity field decreases because the particle velocity is essentially the fluid velocity. One solution is to make the coupling implicit and include the local coupling terms directly in the convection substep rather than as a separate substep. The difficulty with this is that three vector fluid equations have to be solved simultaneously. This necessity, in turn, interferes with the nonlinear and modular nature of a monotone fluid dynamics algorithm.

Another possible solution is to solve a different set of momentum equations. One of these could be the equation for the total momentum for which there would be no coupling terms because they canceled each other term by term. The other two equations could be constructed from differences in the momenta between different phases. The equations are still stiff, but it should be possible to get two orthogonal, decoupled equations where the modules implementing monotone convection would require only minimal change. Recovering the actual particle species densities would then involve inverting the linear transformation used to set up the decoupled equations.

Macroparticle representations for multiphase flows present different coupling problems. In this case, calculating the effect of the fluid on the motion of the particles is straightforward, given the surface interactions. To ensure conservation, however, requires changing the fluid momentum in a consistent way and the coupling can still be stiff. When the macroparticle species are small particles, the drag term will bring the particles to the local fluid velocity almost immediately. The difficulty here is that the velocity change of the gas in a cell cannot be determined until all the particles have been processed for that cell. To do the momentum coupling in a conservative way, therefore, requires two passes through all of the particles. The first pass computes the total particle interaction with the gas and the second pass updates the fluid and each of the particles once the overall gas response has been determined. This double-pass process is required only when the mass of particles is significant.

Furthermore, if there are many macroparticles in one cell and few in another, the corresponding momentum change will differ from cell to cell. These large fluctuations are unphysical, as they are caused by the unphysically small number of simulation macroparticles that can be integrated practically. This means that small shocks can even be generated in the computed flow as adjacent cells are accelerated differentially. This problem is due to poor statistics in the particle representation.

11–4. Coupling Multiple Stiff Processes

There are situations in which timestep splitting, in the form described earlier, fails. Even though global implicit coupling can in principle be used for some of these cases, the computational cost may be prohibitive, and the accuracy may be severely limited. In this section, we describe how timestep splitting may be extended to solve problems with multiple interacting stiff processes.

11–4.1. The General Formulation

As described in Sections 11–1 and 11–2, there are two types of situations in which stiffness in more than one term in the equations makes the problem difficult to solve. These are when:

1. rapid transients arise because one or more of the stiff processes are far from equilibrium, and
2. competing stiff processes are operating close to their common, dynamic equilibrium.

Rapid transients occur for numerical reasons at the beginning of many computations because the systems are often initialized far from equilibrium. Rapid transients also occur as a result of physics in the reactive flow. For example, molecular diffusion, chemical reactions, or both may become temporarily stiff when a shock passes. Rapid transients can also be driven by external source or sink terms. True physical transients, with some of the processes far from equilibrium, really need to be solved on the appropriate short time scales to obtain accurate answers. The saving grace is that the situation *is* transient. It will rapidly approach a condition close to equilibrium, so the number of timesteps needed will be moderate although each step may be very small. Except for situations in which a short-timescale phenomenon becomes oscillatory, or in which an unsteady high-frequency source repeatedly drives the system away from equilibrium, rapid variations in the solution die out quickly.

The second type of stiffness is the situation in which individual processes may be stiff, but the composite solution changes slowly. An explicit integration algorithm would require very short timesteps because the individual processes themselves are stiff. When this type of stiffness exists, the solution for a slowly evolving system, close to its composite equilibrium, is difficult to obtain because the integration must be carried out for a relatively long time with many small timesteps.

Figure 11.2 is a schematic showing the evolution of a system with five physical processes: a source term, chemistry, axial diffusion, radial diffusion, and fluid-dynamic convection. These are chosen to represent those processes used in the example given in Section 11–4.2, but they could also represent some generic processes $G_1(\rho) \ldots G_4(\rho)$, where ρ is a generic density that is a function of temperature T. The solution for ρ and T follows a trajectory in the (ρ, T) plane and is indicated in the figure by the sequence of discrete times t_0, t_1, t_2, and t_3. The separation of these times shows how much the complete solution actually changes each timestep. The three loops of vectors show how the sum of the changes due to the individual processes nearly cancels. In steady state, each loop closes perfectly since the solution does not change. This type of problem is generally considered to be a situation

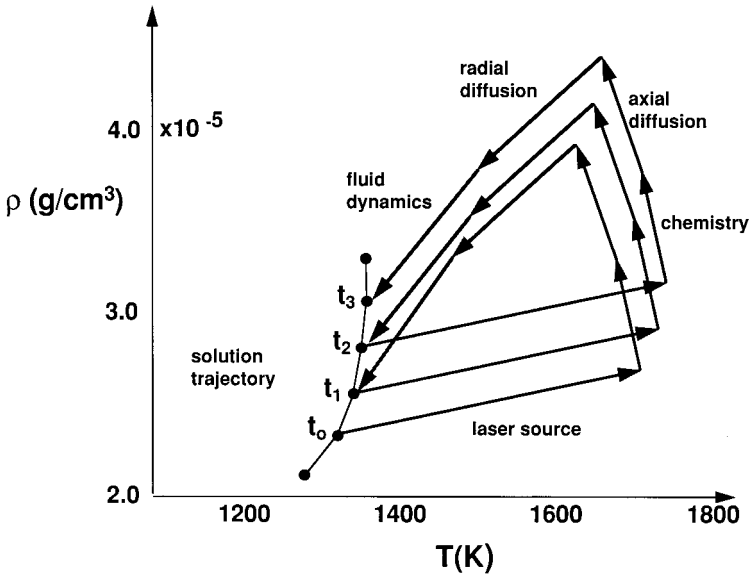

Figure 11.2. Evolution of a multiply stiff system in the phase plane of a radical density and the fluid temperature. The arrows indicate the contributions of the five processes to the solution.

for which timestep splitting does not work, and where global implicit coupling must be used.

The way to treat such systems using timestep splitting (Boris, Oran, and Patnaik 1999; n.d.) is to introduce an estimate of the sum of the effects of the other stiff processes as each stiff process is integrated. In the notation used earlier, the set of physical variables, denoted by $\boldsymbol{\rho}(\mathbf{x}, t)$, is to be advanced using equation (11–1.7). Each of the terms $\{G_j\}$ represents a different physical process or one piece of the direction-split representation of a multidimensional process. Now define the "minus p" operator G_{mp} as the sum of all processes minus process p,

$$G_{mp}(\boldsymbol{\rho}) \equiv \sum_{k=1,\, k\neq p}^{M} G_k(\boldsymbol{\rho}). \tag{11–4.1}$$

Then each process in the global timestep Δt_g is integrated sequentially using

$$\left.\frac{\partial \boldsymbol{\rho}}{\partial t}\right|^{p} = G_p(\boldsymbol{\rho}) + G_{mp}(\boldsymbol{\rho}). \tag{11–4.2}$$

The substep integrations are performed sequentially to evaluate the effects of each term G_p on the system variables. It is important to use the most updated source each time the minus p operator is used.

The operator G_{mp} should be a good approximation to the rate of change of ρ at the current time and position for all processes except p. The two requirements on G_{mp} are that it should (1) incorporate the most recent approximations to the changes in ρ due to various processes, and (2) be able to be evaluated before actually updating the physical variables. These criteria are illustrated in the example in Section 11–4.2. At every substep of the

procedure, it is crucial to enforce the near cancellation of terms that makes the problem stiff. In Figure 11.2, this means that the individual processes are *not* evaluated at the ends of the trail of arrows signifying the composite change associated with all of the previous processes. Instead, all of the processes are considered together, so the physical variables used in the intermediate evaluations are always estimated in a way that they are closer to the final values at the end of the timestep.

In Chapter 8, we described how multidimensional equations could be solved by direction splitting, a particular form of timestep splitting. When the G_{mp} formalism is specialized to solving a multidimensional elliptic or diffusion equation, it is called the alternating-direction implicit (ADI) method (see Chapter 10). In this approach, the solution to the multidimensional problem is obtained by integrating in each direction implicitly, one at a time. The contributions from the other directions, as evaluated at the previous iteration, are used as source terms. Very sophisticated techniques, under the general heading of *dynamic ADI* have been developed to improve the accuracy of this procedure. A main advantage of the ADI approach is that it allows the solutions at different directions to be updated using simpler algorithms, and usually only a tridiagonal matrix solver is required.

Often, the stiff processes are local processes, not requiring the evaluation of a spatial derivative. Coupling such a local stiff process to other processes does not require evaluating additional spatial derivatives, as was discussed earlier in the context of momentum coupling between different phases in closely coupled multiphase flows. Then several of the G_p processes could be allowed to respond simultaneously. In ADI, the nonlocal nature of the derivatives in the directions transverse to the direction currently being integrated are converted to a local term – often at the cost of rigorous conservation. The implicit coupling approach described here, however, should be conservative because the possibly nonconservative approximate terms are canceled after all of the G_i are evaluated.

11–4.2. An Example

Here we use the problem of a laser-driven reaction to illustrate how to use timestep-splitting to solve problems with multiple stiff processes. Consider a flowing gas in which a small volume passes through a region that intersects the beam of a high-intensity laser. The laser adds heat to the gas, and this initiates chemical reactions that generate radicals by dissociating molecules in the gas. When the rate of energy addition to the gas is constant, the region of the disturbance appears as a very narrow, high-temperature plume of atoms in the fluid. These atoms rapidly diffuse into the background gas. In the frame of reference of the fluid moving into the region of the laser energy deposition, the fluid is being rapidly heated and the system may be far from chemical and thermal equilibrium. The fluid has some forewarning that it is going into the region of laser deposition: there is a thermal-conduction wave generated by the fluid element previously heated by the laser. The externally driven expansion of the heated region induces a potential flow that decays rapidly away from the hot spot.

We assume that chemical reactions occur in and near this deposition volume and that the chemical reactions are stiff and endothermic. Because the deposition region is narrow and the gradients are steep, thermal conduction and the molecular diffusion of the newly

created radicals are both stiff in this region. The fluid dynamics is stiff because the sound speed is much greater than the flow speed.

In the frame of reference of the moving fluid, the solution of this problem is changing very rapidly near the laser spot. When the laser intensity is constant, there is a steady-state solution in the frame of reference of the laser. After the initial rapid transients, we expect an expanding, cooling, steady wake behind the laser spot. The important point is that the three stiff processes find a joint equilibrium where thermal conduction, molecular diffusion, and the thermally driven chemical reactions take energy and radicals away from the hot region as the laser adds energy and creates radicals. Fluid convection and the laser source, though not actually stiff here, must also be taken into account. The interaction of these processes is shown schematically in Figure 11.2.

For this problem, equation (11–1.7) can be written as

$$\frac{\partial \boldsymbol{\rho}}{\partial t} \equiv G_R(\boldsymbol{\rho}) + G_Z(\boldsymbol{\rho}) + G_L(\boldsymbol{\rho}) + G_C(\boldsymbol{\rho}) + G_F(\boldsymbol{\rho}), \tag{11–4.3}$$

where the subscript R designates radial diffusion; Z, axial diffusion; L, laser deposition; C, chemical reactions; and F, fluid dynamics. Specifically, there are two scalar equations represented symbolically by equation (11–4.3), one for the radical species ρ and another for the temperature T,

$$\begin{aligned}\frac{\partial \rho\,(r,z,t)}{\partial t} &= \frac{1}{r}\frac{\partial}{\partial r}\left(rD\frac{\partial \rho}{\partial r}\right) + \frac{\partial}{\partial z}\left(D\frac{\partial \rho}{\partial z}\right) + S_L^{\rho}(r,z,t) \\ &\quad + \Gamma(T)\,(\rho_{eq}(T) - \rho(r,z,t)) - V_z\frac{\partial \rho}{\partial z} \qquad (11\text{–}4.4)\\ \frac{\partial T\,(r,z,t)}{\partial t} &= \frac{1}{r}\frac{\partial}{\partial r}\left(rK\frac{\partial T}{\partial r}\right) + \frac{\partial}{\partial z}\left(K\frac{\partial T}{\partial z}\right) + S_L^{T}(r,z,t) \\ &\quad - \frac{\Delta T}{\Delta \rho}\,\Gamma(T)\,(\rho_{eq}(T) - \rho(r,z,t)) - V_z\frac{\partial T}{\partial z}. \qquad (11\text{–}4.5)\end{aligned}$$

The terms $S_L^{\rho}(r,z,t)$ and $S_L^{T}(r,z,t)$ are the laser sources, which are Gaussians with characteristic radius 0.2 cm for ρ and T. The temperature-dependent chemical-reaction rate $\Gamma(T)$ specifies the rate at which the radical density approaches the equilibrium radical density $\rho_{eq}(T)$, which increases exponentially with temperature. The molecular-diffusion coefficient D and the thermal conduction K are constants taken to be large enough that the radial and axial diffusion would be unstable if an explicit algorithm were used. Because of the form chosen for the chemical reaction, the equation for ρ is stiff when $\Gamma(T)$ is large enough. Diffusion is broken into processes in the radial and axial directions, as explained in the previous section, to allow an ADI treatment. Finally, the fluid dynamics is simplified to a constant flow velocity that moves the fluid one cell per timestep toward negative z, so that it only involves a simple shift of the solution along the uniformly spaced z grid.

The problem then has three stiff processes: chemical kinetics, radial diffusion, and axial diffusion. The global timestep Δt_g is determined by the constant flow velocity, V_z, and the spacing of the grid, Δz, such that $\Delta t_g = \Delta z / V_z$. Table 11.4 shows an outline for the numerical model based on the ideas presented above for coupling multiple stiff problems.

Table 11.4. Compensated Operator Splitting for Multiple Stiff Processes

Initialize variables.
∗ Start timestep loop.

1. Replace values (ρ^o, T^o) by (ρ^n, T^n) computed during previous step
2. Find provisional chemistry sources $G_{\rho,c}(\rho^o, T^o)$ and $G_{T,c}(\rho^o, T^o)$ by
   a. Using $G_{\rho,c}$ and $G_{T,c}$ stored from previous timestep, or
   b. Integrating the chemistry one step and backing out the provisional sources,

$$\text{that is, } G_{\rho,c}(\rho^o, T^o) \equiv \frac{\rho^n - \rho^o}{\Delta t} - G_{\rho,mc}(\rho^n, T^n) \quad \text{(ignore updated } \rho^n\text{)}$$

3. Find provisional axial diffusion sources $G_{\rho,z}(\rho^o, T^o)$ and $G_{T,z}(\rho^o, T^o)$ by
   a. Evaluating the diffusion equation explicitly (it was implicit for the step 2) or
   b. Integrating axial diffusion one step and backing out the sources,

$$\text{that is, } G_{\rho,z}(\rho^o, T^o) \equiv \frac{\rho^n - \rho^o}{\Delta t} - G_{\rho,mz}(\rho^o, T_o) \quad \text{(ignore updated } \rho^n\text{)}$$

4. Find the radial diffusion sources $G^r_\rho(\rho^o, T^n)$ and $G^r_T(\rho^o, T^n)$ by integrating radial diffusion one step, using the previously computed sources for this step, and backing out the radial diffusion sources,

$$\left.\frac{\partial \rho}{\partial t}\right|_c = G_{\rho,c}(\rho, T) + G_{\rho,z}(\rho, T) + S_{\rho,l}(\rho, T) + G_{\rho,r}(\rho, T)$$
$$G_{\rho,r}(\rho, T) \equiv \frac{\rho^n - \rho^o}{\Delta t} - G_{\rho,c}(\rho, T) - G_{\rho,z}(\rho, T) - S_{\rho,l}(\rho, T) \quad \text{(ignore } \rho^n\text{)}$$

5. Find the total diffusion sources $G_{\rho,rz}$ and $G_{T,rz}$ by integrating axial diffusion one step using the previously computed sources and saving the total diffusion sources,

$$\left.\frac{\partial \rho}{\partial t}\right|_z = G_{\rho,c}(\rho, T) + G_{\rho,r}(\rho, T) + S_{\rho,l}(\rho, T) + G_{\rho,z}(\rho, T)$$
$$G_{\rho,rz}(\rho, T) \equiv \frac{\rho^n - \rho^o}{\Delta t} - G_{\rho,c}(\rho, T) - G_{\rho,z}(\rho, T) - S_{\rho,l}(\rho, T) \quad \text{(ignore } \rho^n\text{)}$$

6. Integrate the chemistry processes to update the physical variables,

$$\frac{\partial \rho}{\partial t} = \frac{\rho^n - \rho^o}{\Delta t} - G_{\rho,c}(\rho, T) - S_{\rho,l}(\rho, T)$$

   (back out the chemistry sources, if saving them for the next timestep)
7. Integrate fluid dynamics (convection), finding ρ^n, T^n

Start new timestep (go to ∗ above).

Figure 11.3a,b shows gray-scale contours of ρ and T for a case with constant laser intensity. Figure 11.3c,d shows the analogous solutions at one time for a sinusoidally oscillating laser intensity. Each figure compares the solution for two different grid resolutions. On the left is the solution for a fine computational grid, and on the right is the numerical solution at the same time and parameters for a grid ten times coarser. The individual cells can be seen in the coarse computations because no interpolation is used to smooth the solutions.

Parameters for the two computations on the different grids are given in Table 11.5. This table also lists the stiffness factors for the chemistry and diffusion terms. These factors

Table 11.5. Key Parameters for Test Simulations Using Timestep Splitting

Grid	N_R	Δr(cm)	N_Z	Δz(cm)	Δt	$\frac{D\Delta t}{\Delta r^2}$	$\frac{D\Delta t}{\Delta z^2}$	$\frac{K\Delta t}{\Delta r^2}$	$\frac{K\Delta t}{\Delta z^2}$
Fine	300	0.0067	750	0.0133	1.5	30	7.5	15	3.75
Coarse	30	0.0667	75	0.1333	15	3.0	0.75	1.5	0.375

N_R, N_Z – number of computational cells in radial and axial direction, respectively
Δr, Δz – cell size, radial direction and axial directions, respectively
$\Gamma\Delta t$ – stiffness factor for the chemical reactions
$D\Delta t/\Delta r^2$, $D\Delta t/\Delta z^2$ – stiffness factor for radial and axial diffusion, respectively
$K\Delta t/\Delta r^2$, $K\Delta t/\Delta z^2$ – stiffness factor for radial and axial conduction, respectively

are different for the temperature and density because different thermal-conduction and molecular-diffusion coefficients were used. They are also different for each direction for both diffusion terms because the radial cells are half the size of the axial cells. The chemistry becomes less stiff as the resolution is improved because the global timestep decreases with increased resolution. The diffusion terms, however, become stiffer as the resolution increases because the cell size appears quadratically in the denominator of the timestep needed for stability. In both the steady-state and time-dependent problems, the solutions are quantitatively similar despite the factor of ten difference in spatial resolution. At both resolutions, all terms are stiff and some of them are very stiff. These tests show the power of the compensated operator-splitting approach.

11–5. More-Complex Spatial and Temporal Coupling

Coupling representations over disparate scales is a very generic but practical problem that has been dealt with both mathematically and computationally for many years. The first attempts were in terms of solving coupled sets of ordinary differential equations usually

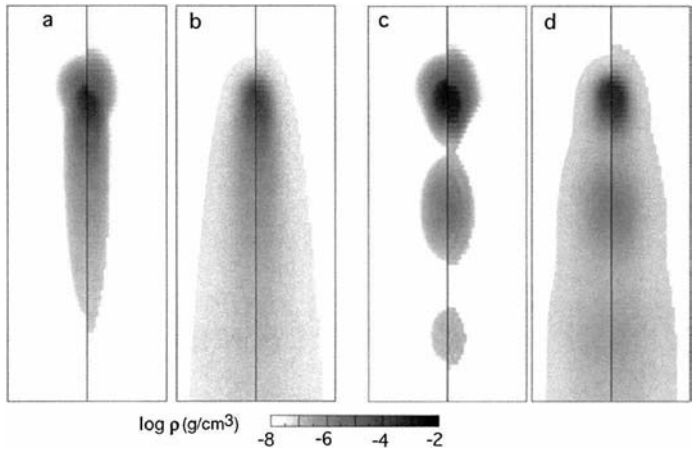

Figure 11.3. Panels (*a*) and (*b*) show gray-scale contours of ρ and T, respectively, for a steady-state solution. Panels (*c*) and (*d*) show the analogous solutions at one time for a sinusoidally oscillating laser intensity. Each panel compares two different resolutions; the coarse-grid calculation is on the right.

representing the evolution of a set of chemical species. When the timesteps required to resolve the fast scales were orders of magnitude smaller than the slow evolution of the solution, this analysis led to solution methods for stiff equations (see Chapter 5). Here we present a more comprehensive discussion and classification of multiscale coupling problems, and describe some advanced techniques that are at the forefront of current reactive-flow and fluid-dynamic research.

11–5.1. A Classification of Time and Space Scales

In Chapter 1 we described the wide range of time and space scales in reactive-flow problems in terms of the different types of mathematical and physical models that become valid as microscopic processes are averaged over successively larger space and time scales. Table 1.1 lists a hierarchy of mathematical models used to describe the behavior of systems involving many particles and interactions. The table shows that the appropriate model can be very different on different scales. The solution methods range from very detailed solutions of fundamental equations describing interactions of particles (such as molecular-dynamics methods), to approximations of systems in which the individual particles are replaced by continuum fluid elements (such as the Navier-Stokes equations). In between these extremes, there are various statistical particle-based methods suited to specific problems. Table 1.1 presents the standard way of describing problems according to the relevant ranges of time and space scales.

In some complex, multiscale problems, the more fundamental models and small scales needed to describe one region or time and space may not be needed everywhere. Finding efficient, or even possible solutions of such problems involves more complex coupling algorithms. An example of such a complex system would be high-temperature, thin-film deposition in which the background flow can be described as an extension of the Navier-Stokes equations, but the reactive deposition process requires a quantum molecular-dynamics approach. In some cases, there is still no model that adequately describes one of the intermediate regimes. As an example, consider the micron-scale regimes in condensed-phase explosives where frictional and interaction stress in the "flow" of these granular materials can only be evaluated from indirect measurements and fit to phenomenological or empirical models.

Another way to classify multiscale problems requires consideration of three general *dimensions of complexity* that help describe the problem. These dimensions represent the degree of:

1. *inhomogeneity* of different spatial and temporal scales,
2. *multidisciplinarity* of the problem, and
3. *discontiguity* of the important scales.

Considering the attributes of a system in terms of these dimensions allows us to focus on the numerical coupling. Figure 11.4 puts these dimensions on a graph, and Table 11.6 shows how various types of reacting flows fit into these classifications, which are now described in more detail.

The first of these classification dimensions concerns the spatial and temporal inhomogeneity of the interacting multiscale processes. *A multiscale phenomenon is homogeneous*

Table 11.6. Another Way to Classify Multiscale Processes

System	Inhomogeneity	Multidisciplinarity	Discontiguity
Ideal turbulence in a box	0	0	0
Thin film reactor	10	10	7 – 10
Porous-flow reactor	3	7 – 10	10
Boundary layer on aircraft wing	10	0	7 – 10
Combustion in automobile engine	0	0	1 – 2
Shock wave	10	0	10

when all the important scales present, microscopic through macroscopic, occupy the same space simultaneously. Fluid turbulence is a good example of a homogeneous phenomenon. There are many scales of motion present at the same time and in the same region of space: the small scales of motion are embedded in the larger-scale eddies. A reactive-flow is inhomogeneous if the different phenomena of interest are widely separated in space or time. One example of an inhomogeneous problem is the deposition of a thin film onto a substrate, as would occur in many types of chemical reactors. The controlling molecular and atomic surface physics is restricted to a narrow domain near the film boundary, while the flow physics occurs on macroscopic scales in the region away from the film. Another inhomogeneous problem is a propagating laminar flame in a premixed gas. For inhomogeneous systems, the additional cost of calculating fine-scale phenomena is at least partially offset by the small size of the region where they are active.

Multiscale phenomena can also be inhomogeneous in time. An example of this is the nonequilibrium response of a macroscopic volume of gas intermittently subjected to short laser pulses. The key question for placing a system on the *Inhomogeneity* axis of Figure 11.4 is: *To what degree are the controlling multiscale physical processes happening in different spatial locations and at different times?* Obtaining an accurate solution with modest computer resources requires us to exploit the inhomogeneity, usually at the cost of program complexity, by using methods that are matched locally to the particular properties of the region in which they are applied.

The second dimension of multiscale classification is the extent to which the problem is multidisciplinary in the sense that the physics and the equations describing the phenomena

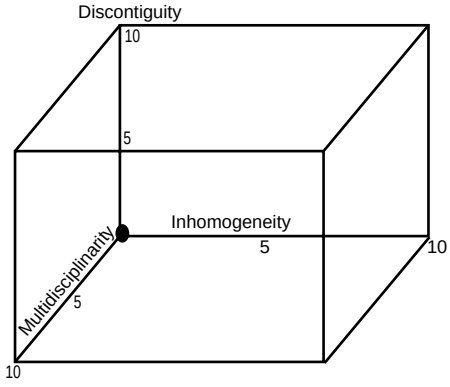

Figure 11.4. Schematic of an alternate way of categorizing reactive-flow problems characterized by multiple time and space scales. Table 11.6 shows where several different types of problems might fit onto this diagram.

at the microscopic and macroscopic scales are quite different. An example of this is diamond growth through chemical vapor deposition (which is also an inhomogeneous multiscale process). The physics of the microscopic diamond growth on the substrate is very different from the macroscopic physics of the large-scale flow throughout the reactor volume. Fluid turbulence, by way of contrast, is not multidisciplinary even though it is multiscale. Multidisciplinary problems, particularly multiscale multidisciplinary problems, are among the hardest to solve. In addition to the multiscale computational complications and the need to merge software of very different types for their solution, there is also the underlying need to merge very different kinds of expertise. The key question for placing a system on the *Multidisciplinarity* axis of Figure 11.4 is: *Is it necessary to use different sets of equations to describe the controlling physical phenomena in different regions for different scales?*

The third dimension of classification describes the degree of contiguity of the actual sizes of the important space and time scales. How separated or contiguous are the relevant scales at the same point in $(\mathbf{x}, t)$? The classical solution procedures described earlier work best when distinct and limited scales are involved. When phenomena are discontiguous, there are intermediate scales between the important large scales and small scales on which not much is occurring. When these intermediate scales are enough smaller than the large scales, and enough larger than the small scales, the existence of the separation can be used to couple the large and small scales. The boundary layer on an aircraft wing is an example of a discontiguous phenomenon. In the boundary layer (particularly the laminar sublayer), the external flow can be treated as an external boundary condition. There are many boundary-layer models based on this separation of scale.

Turbulence, on the other hand, is a good example of a contiguous-scale phenomenon. The scales are contiguously populated with each scale strongly coupled to both larger and smaller scales. Porous flow is an example of a separated-scale (discontiguous) phenomenon that is homogeneous. In porous-body flows, the scale of the pores is generally limited in size, for example, fifty microns to a millimeter for the gaps in sea sand. The macroscales are much larger, for example, ten to one hundred meters for sand bars in the ocean. We expect that once a suitable macroscopic description for porous flow in sand is developed, it will apply quite accurately from the centimeter scale on up. The key questions for placing a system on the *Discontiguity* axis of Figure 11.4 is: *Are the controlling scales of interest, even though they may occupy the same volume, widely separated in size?*

The general problem of developing tractable models that predict the interaction of physical phenomena over a wide range of scales is so broad that a single approach is not likely to evolve. Rather, there will have to be a number of new techniques, some conceptual, some mathematical, and some computational, tailored to specific classes of problems.

11–5.2. Intermittent Embedding: An Advanced Coupling Problem

We end this chapter by discussing a class of problems that require coupling solutions by intermittently embedding one type of solution into another. This could mean embedding a microscopic problem computed by molecular dynamics into a macroscopic solution computed by solving the Navier-Stokes equations, or perhaps embedding a direct-simulation Monte Carlo (DSMC) solution describing a small part of the computational domain into a

Navier-Stokes solution of the entire flow. In these and similar cases, the embedded model is too expensive to solve continuously over the entire computational domain, but the results of the embedded model are necessary to obtain the correct solution. Such problems are generally multidisciplinary, inhomogeneous, and discontiguous. They require coupling physical phenomena that are characterized by very disparate physical scales that most often require very different solution methods. Computing the transfer of information between the smallest and the largest scales of a complex physical system is one of the most difficult and challenging problems in reactive flows.

The usual approach to dealing with such disparate phenomena is to separate the different parts of the problem completely. An example of this would be computations of the properties of the earth's atmosphere at a pole, which requires separately computing atomic processes, such as ionization due to the collision of electrons in air, and the fluid-dynamic processes, such as the local motions of the atmospheric winds. Even though these are, in fact, connected processes occurring simultaneously, the computations could be done separately and the effect of one on the other examined. Another example is the computation of the properties of a chemical reactor designed to grow layers of thin films on a substrate. The motions of material through the reactor may be primarily a reactive-flow problem, but the deposition process that forms the film is an atomic or molecular problem of surface chemistry and physics. For both the atmospheric and the reactor problems, combining the effects that occur on the disparate scales is usually done with phenomenological models implemented through input data, initial conditions, or boundary conditions.

Such approaches to dealing with the separation of scales results in communities with very different approaches, algorithms, instrumentation, diagnostic methods, and, effectively, very different languages. Now we are at the point where these disciplinary separations are no longer adequate. We need models that allow more accurate, closely coupled transfer of information among the scales. Thus we must explore modifications to standard approaches as well as invent new ways of combining the information we have about complex, multiscale physical systems.

Consider a model of the thin-film reactor that produces diamond films using the techniques of chemical vapor deposition. Suppose that the flow in this reactor can be physically divided into two spatial domains: (1) the hot gas flowing in the reactor that is away from the surface, and (2) the growing surface and the very thin layer of gas near and at the surface. The diamond is growing through fast surface-activated chemical reactions occurring in an environment of very steep temperature and density gradients. The two regions are closely coupled because the reactive flow responds to the heat and density changes originating at the surface. The effects of changes in the surface conditions are only slowly felt in the gaseous flow. The physics and chemistry of the two regions are governed by different types of interactions that must be described by totally different sets of equations: Navier-Stokes equations for the general flow through the reactor, and molecular dynamics for the diamond growth process. This problem is multidisciplinary, inhomogeneous, and discontiguous.

Coupling the disparate time and space scales could be done by an extension of the concepts of timestep splitting described above. For example, we might consider solving the molecular dynamics model for a fixed amount of time, and then using this information as input to the fluid solution. This would mean that a microscopic model has been embedded in a macroscopic model. Because the timestep for the molecular-dynamics model is very

short and the computation itself is expensive, it is impractical to solve the two models simultaneously for very long. Therefore we also need a connecting surrogate model that synthesizes the results of the microscopic behavior and can be used as part of the reactive-flow model during the long periods when the microscopic model is turned off. The short periods when the microscopic model is solved are used to update the parameters of the surrogate model based on the current state of the macroscopic flow.

The issue then is to solve the embedded, microscopic model intermittently for only a few selected locations or situations occurring in the large-scale simulation. This information may then be used to drive a phenomenological model that provides input to the macroscopic model everywhere describing the behavior of the unresolved phenomena. Questions to be addressed are: How often, and at how many points, and for how long in physical time must the embedded model be solved? How can the macroscopic phenomenology be updated *automatically*? An important part of this problem is developing a general algorithm that will control how much of the microscale process must be simulated to get reasonable quasistatic results to use in the macroscale computation. It is necessary to develop accurate and stable, but problem-dependent coupling criteria.

In the reactor problem, the growth of the surface is always on or just above the fixed location of the substrate surface. Computations with moving embedded surfaces, such as flames or freezing, have an additional difficulty over that of the diamond reactor because the location of the embedded model moves self-consistently in the flow. Now the macroscale model would describe fluid flow and heat transport to and away from the interface. The microscale model would consist of an expanded representation of the physics and chemistry at the surface. An overall modeling issue would be to track the interface motion using an easily computable macroscopic representation and then couple this to both the microscopic model and the multidimensional fluid-dynamics model. Chapter 6 described ways that this adaptive embedding could be accomplished geometrically.

There have been some current efforts to combine results of particle-based simulations, such as Monte Carlo and molecular-dynamics methods to continuum fluid solutions. This problem has been discussed and summarized to some extent by Oran, Oh, and Cybyk (1998), who described various efforts and problems connected with combining DSMC, Navier-Stokes, and molecular-dynamics solutions. Some of the problems connecting Navier-Stokes and Monte Carlo solutions involve how to maintain continuity of thermodynamic and transport properties across the interfaces between two descriptions. Issues involved in coupling molecular dynamics and Navier-Stokes simulations involve how to set up the boundary conditions and deal with timesteps that vary from femtoseconds to microseconds or even milliseconds. These are now forefront issues in the computation of reactive flows.

REFERENCES

Boris, J.P., E.S. Oran, and G. Patnaik. n.d. Convergence and accuracy of compensated operator split coupling of multiple stiff processes. *Journal of Computational Physics*.

Boris, J.P., E.S. Oran, and G. Patnaik. 1999. Coupling multiple stiff processes in reactive flow simulations. In *17th International Colloquium on the Dynamics of Explosions and Reactive Systems* (ICDERS). Heidelberg, Germany.

Dean, E.J., and R. Glowinski. 1993. On some finite element methods for the numerical simulation of incompressible viscous flow. In *Incompressible computational fluid dynamics*, 17–56. New York: Cambridge University Press. M.D. Gunzburger and R.A. Nicolaides eds.

Douglas, J. 1962. Alternating direction methods for three space variables. *Numerische Mathematik* 4: 41–63.

Douglas, J., and M. Rachford. 1956. On the numerical solution of the heat conduction problem in two and three space variables. *Transactions of the American Mathematics Society* 82:421–439.

Gamezo, V.N., and E.S. Oran. 1997. Reaction-zone structure of a steady-state detonation wave in a cylindrical charge. *Combustion and Flame*. 109:253–265.

Gardner, J.H., S.E. Bodner, and J.P. Dahlburg. 1991. Numerical simulation of ablative Rayleigh-Taylor. *Physics of Fluids* B3:1070–1074.

Leveque, R., and J. Oliger. 1983. Numerical methods based on additive splitting for hyperbolic partial differential equations. *Mathematics of Computation* 40:469–497.

Marchuk, G.I. 1975. *Methods of numerical mathematics*. New York: Springer-Verlag.

Oran, E.S., and C.R. DeVore. 1994. The stability of imploding detonations: Results of numerical simulations. *Physics of Fluids* 6:1–12.

Oran, E.S., C.K. Oh, and B.Z. Cybyk. 1998. Direct simulation Monte Carlo – Recent advances and applications. *Annual Reviews of Fluid Mechanics* 30:403–441.

Peaceman, D., and M. Rachford. 1955. The numerical solution of parabolic and elliptic differential equations. *SIAM Journal* 3:28–41.

Strang, G. 1968. On the construction and comparison of difference schemes. *SIAM Journal of Numerical Analysis* 5:506–517.

Ton, V.T., 1996, Improved shock-capturing methods for multicomponent and reacting flows, *Journal of Computational Physics* 128:237–253.

Weber, J.W., Jr., E.S. Oran, J.D. Anderson, Jr., and G. Patnaik. 1997. Load balancing and performance issues for the data parallel simulation of stiff chemical nonequilibrium flows. *AIAA Journal* 35: 486–493.

Williams, D. 1997. Private communication. Method used in D.N. Williams, L. Bauwens, and E.S. Oran. 1996. A numerical study of the mechanisms of self-reignition in low-overdrive detonations. *Shock Waves* 6:93–110.

Yanenko. N.N. 1971. *The method of fractional steps*. New York: Springer-Verlag.

12

Turbulent Reactive Flows

Any attempt to define turbulence in a few words, or even a few lines, would probably invite argument and cause confusion. Turbulence is best described by a few of its characteristics. Turbulent flows are generally high Reynolds-number flows that appear to be irregular or random. Turbulent fluid motions are complex and contain many different time and space scales all coexisting in the same volume of fluid. In the terminology used in Section 11–5.1, turbulence is generally a homogeneous phenomena in the sense that all of the important scales present, microscopic through macroscopic, occupy the same space simultaneously, and it is contiguous in the sense that the relevant spatial and temporal scales are very close or overlapping. Experiments on turbulent flows are not microscopically reproducible from one time to the next.

Perhaps the most important aspect of turbulence for reactive flows is that it provides an efficient way for distinct, initially separate materials to interpenetrate and mix. Turbulence greatly increases the rates of heat, mass, and momentum transfer, as well as interspecies mixing, which is usually a necessary precursor for chemical reactions. This rapid mixing is caused by the spectrum of vortices in the flow, which act to increase the surface area of the interface between different and partially unmixed materials. As the interface surface area increases, proportionately more material diffuses across this interface, so that more molecular-scale mixing occurs. Therefore, a turbulent flame with its convoluted surface area propagates faster than a laminar (nonturbulent) flame because of the resulting faster energy release. On the computational side, the addition of chemical reactions and heat release makes it more expensive to simulate reacting turbulent flows than nonreacting turbulent flows. Nonetheless, there has been substantial progress in this area in the last ten years because of improved theoretical understanding that has resulted from both analysis and simulations.

Turbulence also has important effects on boundary layers. In the aerodynamics community, there is a tendency to equate, or at least confuse, boundary layers and turbulence. This is understandable because boundary layers are the source of the turbulence in typical aerodynamic problems in which a body moves through a uniform, nonturbulent fluid. In fact, boundary layers are actually a fluid-structure interaction, not a purely fluid-dynamic effect, and are driven at small scales, unlike turbulent cascade. Boundary layers are quite laminar close to the body and become turbulent because of the strong vorticity being shed

by the body. Turbulence in the external flow modifies boundary layers, and the presence of the body, acting through the boundary layer, modifies turbulence.

This chapter describes some of the concepts and definitions that form our current picture of turbulence, summarizes selected aspects of the various approaches to computing nonreactive turbulent flows, and then describes computations of turbulent reacting flows. Computation is more and more becoming the methodology of choice for studying complex, nonlinear flows. The material presented here in no way replaces a course or a book on turbulence, such as Tennekes and Lumley (1972), Hinze (1975), or Libby and Williams (1980, 1994). The approach to turbulence presented here is somewhat different, as it comes from a computational perspective. The volumes of the *Annual Reviews of Fluid Mechanics* of the last ten years contain several excellent review articles that are referenced throughout. The book *Whither Turbulence? Turbulence at the Crossroads* (Lumley 1990) is a summary of a conference that focused on and discussed current concepts.

The underlying themes of this chapter can be summarized by two questions:

> *Given a turbulent reactive-flow problem, how much of the wide range of time and space scales must be resolved computationally to obtain an adequate answer?*

and then

> *How can we use the general information we have about the behavior of a turbulent flow, including the information about the relevant scales, to produce an adequate computation?*

12–1. Concepts for Modeling Nonreactive Turbulence

Historically, any flow that had the appearance of being random was called turbulent. Thus regular but very complex flows were sometimes called turbulent. We now know that flows that were thought to be statistically random are dominated by the persistence of relatively large structures. These "coherent" structures were previously ignored because their existence was largely masked by experimental averaging techniques designed to measure the expected statistical properties of turbulence. The classical descriptions and analyses of turbulence do not adequately explain and account for the effects of these unsteady but organized and persistent coherent structures.

12–1.1. An Overview of Current Concepts and Approaches

The classical descriptions of turbulence evolved from noting that many flows exhibit a series of rather sudden transitions to new flow patterns as the Reynolds number is increased. Each transition results in an increasingly complicated flow. At a sufficiently high Reynolds number, the flows become irregular and appear random in both space and time. The transitions may be caused by a sequence of fluid instabilities, each of which introduces more structure (often by breaking a symmetry) in a previously stable or periodic flow and thus introduces new, smaller scales into the flow. The transition to a turbulent flow was postulated to occur through an infinite succession of instabilities, each contributing to the

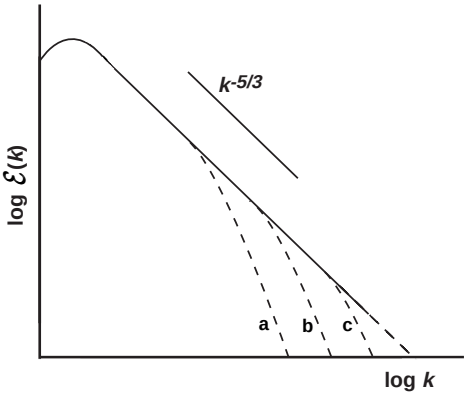

Figure 12.1. Kolmogorov spectrum showing the decay of the turbulent kinetic energy $\mathcal{E}$ with wavenumber k. The dashed lines show how the curve changes with decreasing Reynolds number (a) lowest Re, (b) higher Re, and so on.

increasingly high frequency of the motions in the flow (Landau and Lifshitz 1959). Today we know that systems may often become turbulent after a sequence of only a few instabilities at incommensurable scales. Statistical theories of turbulence were largely derived to describe fluctuations in macroscopic properties such as velocity, density, and temperature.

In a statistically steady, turbulent flow, the energy density in the velocity fluctuations is denoted by $\mathcal{E}(k)$, where the length scale $1/k$ corresponds to the wavenumber k. For a three-dimensional flow, $\mathcal{E}(k)$ follows a power-law spectrum that decays at small scales according to a $k^{-5/3}$ power law, as shown in Figure 12.1. This means that most of the energy in the flow is contained in the large scales, not the small scales. The scales of motion range from the system size L, down through intermediate scales in an *inertial range* that follows this $k^{-5/3}$ decay, to a small *Kolmogorov scale* l_K. At approximately l_K, the effects of viscous dissipation are to convert the organized motions of the small vortices into heat, creating a sink of kinetic energy at scales near l_K. Most of the energy associated with turbulence cascades from the large scales into the smaller scales and there is relatively little feedback from the small to the large scales. Turbulence is *isotropic* and *homogeneous* when its statistical properties are largely independent of direction and position. (Note that the definition of "homogeneous" used here is different from the one used to describe scale separation in Section 11-5.)

In the current picture, a substantial amount (more than half) of the vorticity and energy in a turbulent flow exists in the form of these *coherent structures*. These structures exist on scales ranging approximately from L to l_K and they consist of relatively closely bound vortex filaments that move collectively in the flow. These structures maintain a coherent behavior over a time longer than their rotation time. This picture, that coherent structures dominate flows, is supplemented by the following ideas.

- Turbulent flows are not homogeneous and isotropic, even though some turbulence created in a laboratory comes close. Such a state is only a limiting condition.
- Since coherent structures are well documented on large and intermediate scales, similar structures almost certainly exist on all scales larger than the dissipation scale.
- The practical transition to turbulence in *transitional flow* does not seem to occur through an infinite succession of instabilities, but occurs more abruptly after a sequence of relatively few instabilities, typically two or three.

- *Intermittency* is a situation in which a flow, usually considered to be turbulent globally, is characterized by several identifiably different and distinct states. Each of the states has a statistical probability of being found in a particular region, but there is no predictable time of recurrence. One example of intermittency is an overall laminar flow with finite-sized turbulent patches convected along with it. Another example is the opposite case of a turbulent flow with pockets of irrotational fluid. The existence of intermittency in turbulent flows indicates the presence of relatively narrow interfaces separating regions of fluid having distinctly different properties. The occurrence and importance of intermittency with its sharp interfaces means that this aspect of turbulence cannot be well represented as a fundamentally diffusive effect.

These ideas have been examined in experiments, theories, and extensive numerical simulations of turbulent and transitional flows.

Many initially laminar flows undergo a transition to turbulence. Transitional flows may not yet have reached the equilibrium state implied by the Kolmogorov cascade in Figure 12.1. Further, because of dynamic processes, nominally turbulent flows may be driven away from equilibrium. Thus, important classes of turbulent flows, especially those in reactive systems, are not in equilibrium and cannot be characterized by the energy spectrum shown in Figure 12.1. Recently, such flows have been called *nonequilibrium turbulent flows* or *non-Kolmogorov turbulent flows*.

Because of the practical importance of turbulence in changing mixing-layer and boundary-layer characteristics, and its prevalence in natural flows, there has been an extensive international effort to visualize, measure, and compute its properties. A generally held, fundamental assumption of the computations, and of turbulence research generally, is that all of the physics of turbulence is contained in the Navier-Stokes equations. Thus if we could resolve all of the scales in a problem from L to l_K, we could compute the properties of the flow completely. Resolving the complete range of scales in a computation is called *direct numerical simulation* or *DNS*. Understanding and predicting turbulent flows is so important that the need for DNS has justified the development and use of the world's largest supercomputers.

Unfortunately, even the fastest and largest computers are neither fast enough nor large enough to solve most turbulence problems from first principles using the Navier-Stokes model. The fundamental difficulty is expressed in the two questions at the beginning of this chapter, which can be asked another way: How do we deal with the vast disparity in time and space scales among the fundamental physical processes? Because the understanding of and eventual ability to predict the behavior of turbulent flows is imperative for many real applications, we need some practical, reasonably accurate algorithms that will adequately describe the effects of these widely disparate scales in a single computation. The enormous computational requirements of straightforward DNS have led to a number of more approximate approaches.

One approach is to use the Navier-Stokes equations to compute the properties of the larger, energy-containing scales in the flow, and then model the behavior of the smaller, unresolved scales with a *subgrid turbulence model*. This process is called *large-eddy simulation* or *LES*. A major part of LES research is determining how to couple the subgrid model to the large-scale computed flow. The difficulty is that the scales are contiguous,

and the connection between the computed and modeled scales must be made smoothly. Relatively recent efforts, focused on examining the extent to which subgrid models are required, have led to the concept of *monotone-integrated LES*, or *MILES*. The observation underpinning this approach is that a wide class of modern CFD algorithms mimics the behavior of the physical flow down to the grid-scale cutoff, where the algorithm then dissipates the structures in a way similar to viscous dissipation at l_K. When the interest is in the large energy-containing scales, it is unnecessary to resolve the flow down to l_K. LES and MILES are discussed in Sections 12–2 and 12–4, respectively.

Another approach replaces the Navier-Stokes equations with a set of averaged flow equations called the *Reynolds-averaged Navier-Stokes* (or *RANS*) equations. Here a formal averaging procedure results in a hierarchy of equations that require models to describe averages of products of fluctuations used to complete or "close" the equations. Extensive efforts have gone into closing the equations, which requires deriving appropriate *turbulence closure models*, often simply called *turbulence models*. Key elements of these models are parameters obtained both by fitting solutions of the equations to experimental data and from detailed computations. Both experiments and DNS have played an important part in deriving and calibrating turbulence closure models.

12–1.2. Direct Numerical Simulation

DNS uses the full power of a computer to solve the Navier-Stokes equations for the properties of a turbulent flow over the complete range of scales determined by the physics of the problem. The computational domain must be large enough to include the largest scales, and the mesh spacing must be fine enough to resolve the smallest scales. Limitations on computer speed and memory have meant that direct simulations can only be done for a limited class of low–Reynolds-number problems in idealized configurations. For example, DNS can now be performed rather routinely using $512 \times 512 \times 512$ grids with Reynolds numbers approaching a thousand or more. Such calculations will continue to be expanded as faster computers with larger memories become available.

DNS is a fluid-dynamics research tool for studying turbulence, not an approach for solving the Navier-Stokes equations for engineering problems. In DNS, turbulence is a solution predicted at great expense from the contributing physical processes, not a separate process captured inexpensively, as is the case when turbulence subgrid models are used. This point is made definitively in many articles, including the review of DNS by Moin and Mahesh (1998). Therefore, DNS must be viewed as different from, but complementary to, LES and turbulence models. DNS is best used to uncover important properties of a flow, to test theoretical predictions, and as an idealized experiment from which to calibrate the physical relationships built into various LES and turbulence models.

Different types of time-dependent CFD representations are used for DNS of turbulent and transitional flow. These include: spectral representations, grid-based representations (such as finite-difference, finite-volume, and finite-element algorithms), and even vortex-dynamics representations. Sometimes computations may use a combination of these methodologies. DNS became an important tool for studying turbulence shortly after the invention of the fast Fourier transform (FFT) algorithm (Cooley and Tukey 1965). The earliest work in DNS, done for atmospheric flows, was reviewed by Fox and Lilly (1972).

DNS computations of isotropic turbulence began in the early 1970s (see Orszag and Patterson [1972]), evolving in the 1980s to computations of wall-bounded turbulence (e.g., Rogallo [1981], and free shear layers (see, e.g., Riley and Metcalfe [1980]). In the past ten years, DNS has been used for more complicated flows, such as flows over walls and steps, although the Reynolds numbers are still low. The types of problems computed have included the evolution and decay of turbulence, turbulent flow in a channel, the evolution of mixing layers in free and bounded shear flows, shock-turbulence interactions, and the structure of turbulent boundary layers. We particularly recommend the reviews by Rogallo and Moin (1984), Kleiser and Zang (1991), and Moin and Mahesh (1998).

DNS for compressible flows was started in the early 1980s with solutions for homogeneous compressible turbulence (Feiereisen, Reynolds, and Fergizer 1981), and evolved to isotropic and sheared compressible homogeneous turbulence (e.g., Erlebacher et al. [1990] and Blaisdell, Mansour, and Reynolds [1993]) and now turbulent boundary layers (Rai, Gatski, and Erlebacher 1995). High-speed turbulent mixing layers have been computed for flows in the range at $Re = 50 - 150$ by Vreman, Sandham, and Luo (1996). As discussed later, DNS is now becoming a very useful tool for isolating and computing properties of reactive flows.

Perhaps the simplest turbulence problem considered computationally is the decay of turbulence in a cubic domain, with periodic boundary conditions on all sides. The flow may be driven at the largest scale in some artificial way, or initialized and allowed to decay in time. This configuration is argued to represent a small region embedded in a much larger system and far from the boundaries. This problem has been used to study the nonlinear interactions of fluids, the cascade of turbulent energy from the macroscopic to the dissipative scales, and to provide a database for testing subgrid turbulence models. Early simulations used spectral methods to study incompressible, constant-density turbulence. Such computations are continued today at considerably higher resolution and using a variety of spectral, spectral-element, and finite-volume methods. They have also been extended to include the effects of chemical reactions with energy release. This periodic geometry, because of its simplicity, is often used to evaluate new architectures for parallel computers.

12–1.3. Turbulence-Averaged Navier-Stokes Equations

Some approaches to modeling turbulence are based on a statistical treatment of fluctuations about a stationary or very slowly varying flow. The primary variables are broken into two parts: a mean, time- or ensemble-averaged part of the flow, and a fluctuating part representing deviations from this mean. The *Reynolds decomposition* of the system variables can be defined as:

$$f(\mathbf{x}, t) = \overline{f}(\mathbf{x}) + f'(\mathbf{x}, t), \tag{12–1.1}$$

where f indicates, for example, $\mathbf{v}$, P, ρ, n_i, T, or E, where the variables are defined as in Chapter 2. The bar over a quantity indicates the time-averaged value,

$$\overline{f} \equiv \lim_{\Delta t \to \infty} \frac{1}{\Delta t} \int_{t_o}^{t_o + \Delta t} f(t)\, dt. \tag{12–1.2}$$

The prime indicates the fluctuating part which satisfies the condition

$$\overline{f'(\mathbf{x}, t)} = 0. \tag{12–1.3}$$

The strength of this approach is that it isolates unknown, fluctuating parts of the solution so that they may be suitably approximated. This approach is only correct in the limit that the mean flow is stationary, otherwise $\overline{f}(\mathbf{x})$ and $f'(\mathbf{x}, t)$ really cannot be cleanly separated. There are intrinsic errors when these turbulence-averaged approaches and related subgrid-filtering approaches are applied to situations where $\overline{f}(\mathbf{x})$ is a function of time, or the geometry is complex or variable. (In the general case, these errors cannot be eliminated by ensemble or local time averaging.) When time variations are allowed in the mean quantities, there is an assumption made that the variation is slow enough that the separation is at least useful and valid asymptotically because the scales are well separated.

The *RANS* equations are found by taking the mean value of the fluid-dynamic equations. For example, the mass density and momentum equations become

$$\frac{\partial \overline{\rho}}{\partial t} + \nabla \cdot \overline{\rho \mathbf{v}} = 0 \tag{12–1.4a}$$

and

$$\frac{\partial \overline{\rho \mathbf{v}}}{\partial t} + \nabla \cdot \overline{\rho \mathbf{v}\mathbf{v}} = -\nabla \overline{P} - \nabla \cdot \overline{\boldsymbol{\tau}}. \tag{12–1.4b}$$

These two equations, plus the averaged equation for energy conservation (not shown here), are the starting points for most turbulence-modeling and subgrid-filtering approaches. Note that these equations are rigorously correct when the flow is steady, which means that $\partial \overline{\rho}/\partial t = 0$ and $\partial \overline{\rho \mathbf{v}}/\partial t = 0$.

For incompressible flow, the issue of a mean quantity changing in time becomes more obvious. With $\rho' = 0$, $\overline{\rho}$ now must be constant in space as well as time. The only freedom left is in the rotational components of $\mathbf{v}$. Then equation (12–1.4) becomes

$$\nabla \cdot \overline{\mathbf{v}} = 0 \tag{12–1.5a}$$

$$\overline{\rho} \frac{\partial}{\partial t} \overline{\mathbf{v}} + \overline{\rho} \nabla \cdot (\overline{\mathbf{v}}\overline{\mathbf{v}}) = -\nabla \overline{P} + \nabla \cdot (\overline{\boldsymbol{\tau}} - \mathbf{T}). \tag{12–1.5b}$$

These constant density RANS equations have the same general form as the fluid equations in Chapter 2, but now there is an additional term called the *Reynolds stress*, $\mathbf{T}$,

$$\mathbf{T} \equiv \overline{\rho}\, \overline{\mathbf{v}'\mathbf{v}'} = \overline{\rho}\, \overline{(\mathbf{v} - \overline{\mathbf{v}})}\ \overline{(\mathbf{v} - \overline{\mathbf{v}})}. \tag{12–1.6}$$

Much of the work of turbulence modeling is concerned with how to model this term. Eddy-viscosity models (zero-equation models) represent $\mathbf{T}$ in terms of a turbulent viscosity coefficient times derivatives of the mean properties of the flow. Higher-order closure models describe the evolution of $\mathbf{T}$ by a partial differential equation (see, e.g., Hinze [1975] or the review in Speziale [1991]).

Again, the quantities $\overline{\rho}$ and $\overline{\mathbf{v}}$ in equations (12–1.4) and (12–1.5) are defined as time averages although they appear with time derivatives. This apparent inconsistency can be partially justified when $\overline{f}$ and f' are both time dependent, but vary on very different time

scales. In the limit that the time average in equation (12–1.2) extends over an interval that is long compared to the fast turbulent variations in f, this kind of a separation can be justified. This is not generally the case in turbulent flows, since these flows contain a continuous spectrum of scales.

It is always possible to write filtered Navier-Stokes equations, such as equation (12–1.1), with a finite time or space average. Then the core of the problem is to find a formulation in which determining the Reynolds stresses is not too difficult or arbitrary. Because this kind of averaging does not result in a complete separation of scales, the maximum errors are in the scales that are represented by subsequent closure models. Turbulence-averaged approaches inherently contain errors that are difficult to characterize.

For compressible flows where $\rho' \neq 0$, the density equation is now

$$\frac{\partial \overline{\rho}}{\partial t} + \nabla \cdot \overline{\rho}\,\overline{\mathbf{v}} + \nabla \cdot \overline{\rho' \mathbf{v}'} = 0, \tag{12–1.7}$$

which contains the term, $\nabla \cdot \overline{\rho' \mathbf{v}'}$. Difficulties associated with modeling extra terms such as this have led to *Favre averaging*, which defines mass-weighted quantities such as

$$\widetilde{\mathbf{v}} \equiv \frac{\overline{\rho \mathbf{v}}}{\overline{\rho}}. \tag{12–1.8}$$

The velocity may be written as

$$\mathbf{v}(\mathbf{r}, t) \equiv \widetilde{\mathbf{v}}(\mathbf{r}) + \mathbf{v}''(\mathbf{r}, t), \tag{12–1.9}$$

where now $\mathbf{v}''$ is a slightly different velocity fluctuation. This formulation simplifies the resulting density equation by eliminating the extra term and giving a continuity equation with no source terms. The *Favre-averaged Navier-Stokes equation* for density is

$$\frac{\partial \overline{\rho}}{\partial t} + \nabla \cdot (\overline{\rho}\,\widetilde{\mathbf{v}}) = 0. \tag{12–1.10}$$

For the momentum equation it is

$$\frac{\partial}{\partial t}(\overline{\rho}\,\widetilde{\mathbf{v}}) + \nabla \cdot (\overline{\rho}\,\widetilde{\mathbf{v}}\widetilde{\mathbf{v}}) = -\nabla \overline{P} + \nabla \cdot (\overline{\boldsymbol{\tau}} - \overline{\rho \mathbf{v}' \mathbf{v}'}). \tag{12–1.11}$$

For the energy equation, written in terms of the enthalpy, h, the Favre-averaged version of the equation becomes

$$\begin{aligned} \frac{\partial}{\partial t}(\overline{\rho}\,\widetilde{h}) + \nabla \cdot \overline{\rho}\,\widetilde{h}\,\widetilde{\mathbf{v}} = {} & \frac{\partial \overline{P}}{\partial t} + \widetilde{\mathbf{v}} \cdot \nabla \overline{P} + \overline{\mathbf{v}' \cdot \nabla P} \\ & + \nabla \cdot (-\overline{\mathbf{q}} - \overline{\rho h' \mathbf{v}'}) + \overline{\boldsymbol{\tau} \cdot \nabla \mathbf{v}'}. \end{aligned} \tag{12–1.12}$$

Now the Reynolds stresses include the density variation as well. Furthermore, there are additional high-order terms that have to be modeled. Similar approaches applied to turbulent compressible flows with chemical reactions lead to RANS equations with even more time-averaged terms. The computational and algorithmic difficulties are about the same. The major questions are (1) how to close the equations by modeling all of the stress terms? and (2) how much error is made by the averaging assumptions?

To close the set of turbulence-averaged equations requires models for the undefined turbulence quantities, such as the Reynolds stresses and the turbulent heat fluxes in equations (12–1.11) and (12–1.12). This is done with *turbulent scale models*, which are often referred to simply as *turbulence models*.

12–1.4. Turbulence Modeling and Reynolds-Stress Models

A turbulence model provides a description of the connections between the products of turbulent fluctuations in the averaged equations and other variables or parameters available or easily computable. This connection is usually a combination of algebraic and evolution equations relating the turbulence parameters to the averaged dependent variables. To close the set of equations, the Reynolds stress is sometimes modeled as a function of a *turbulent velocity scale*, q, a *turbulent or eddy viscosity*, ν_T, and a *turbulent length scale*, l, where ν_T is a function of q and l. Such models for the Reynolds stresses are classed as zero-equation, one-equation, and two-equation models, or stress-equation models (see, e.g., Reynolds [1976]).

There are several distinct approaches to developing Reynolds-stress models. One is a continuum mechanics approach. This may be based on an expansion, such as a Taylor expansion, and then a series of physical constraints are applied. Key experimental results are used to pin down model constants. Another approach is a statistical-mechanics approach based on an asymptotic expansion. An interesting and satisfying aspect of this approach is that the constants in the model are calculated explicitly, if not always correctly, for a particular problem. Renormalization group (RNG) models are an example of this approach (see Yakhot and Orszag [1986a,b]). It is important to note that both approaches can give zero-, one-, and two-equation models, as well as higher-order closures.

Zero-equation models solve partial differential equations such as equation (12–1.5) for the mean velocity field and use a *local algebraic model* to relate the fluctuating quantities to the mean flow. Models of this type work well enough in simple quasisteady flows for which the adjustable constants have been calibrated. They do not work well for problems in which the mean conditions change abruptly or when a significant region of the flow is affected by turbulent transport. Zero-equation models assume that the turbulence has equilibrated with the local mean conditions. These models do not explicitly consider the evolution histories of turbulence or reacting species in the flow.

One-equation models add a coupled partial differential equation for the turbulence velocity scale. *Two-equation models* add yet another equation to establish the turbulence length scale. Examples of these are the k–ϵ, k–l, or k–ω models. The additional partial differential equations are coupled to the computed mean flow to describe the evolution of the turbulence-related quantities. These, in turn, are used to calculate the stress tensors in the mean flow equations. These models, which have been described as *scale evolution models*, relate turbulent transport to a combination of local features of the mean flow and one or two scalar parameters that can provide a phenomenology for the evolution of the turbulence. Nonetheless, they are still equilibrium models, assuming the mean field and turbulence parameters to be in equilibrium locally. Evolution equation models generally give a reasonable approximation of the turbulence in simple separated flows or flows with gradual changes in boundary conditions.

They often fail in flows with strong rotations, density gradients, and chemical-energy release.

Stress-equation models, also called *transport evolution models*, solve partial differential equations for all of the components of the turbulent stress tensor. These models, derived from the Navier-Stokes equations themselves (with the caveats given above), are based on equations for the evolution of the transport terms in the averaged Navier-Stokes equations. In addition, at least one equation is solved for a parameter describing the turbulence length-scale information. Local algebraic models provide the higher-order statistical quantities in these equations. For example, the *Reynolds-stress transport* models, in which the Reynolds stresses become dependent variables of the system of partial differential equations, are of this type. They are computationally expensive, and using them does not necessarily produce answers substantially better than the simpler algebraic and scalar evolution models. Again, a significant source of error occurs in attempts to solve models that define mean quantities in a time-varying flow. These stress-equation models become even more complex in compressible reactive flows.

Stress-equation models were developed to address some of the fundamental deficiencies in two-equation models. These deficiencies arise because the two-equation models imply a separation of scales at the second moment level and a certain degree of isotropy (Speziale 1991). The result is higher-order closure representations, the simplest of which is the second-order closure. To use these representations, however, requires approximate models for higher-order correlations in the Reynolds-stress equations.

The turbulence-averaged equations, whatever their final form, may generally be solved by the numerical methods described earlier in this book. These equations are often used in situations where the mean flow is nearly steady state, a situation where they should be most accurate. The danger is in using them when the macroscopic flow is dynamic and the computation is poorly resolved. Then the physical effects represented by turbulence models may be masked by numerical diffusion, unless the actual turbulent viscosity is large. Also, there is a question about the interpretation of turbulent quantities in time-dependent problems, such as those involving vortex shedding. An inherent part of turbulence modeling is the multiple-time-scale approximation. This approximation is implicitly invoked, although it is not rigorously implemented and probably not strictly valid.

12–1.5. Probability Distribution-Function Methods

Probability distribution-function (PDF) methods are a statistical approach for treating inhomogeneous turbulent flows. They can be used as an alternate approach to the closure approaches described earlier or as an enhancement to them for treating more complex phenomena such as reacting flows. The PDF allows treatment of a number of distinct configurations contributing to the mean, though phase and subgrid localization information is not available. For this reason, unlike most turbulence-closure models, PDFs can include intermittency effects.

Define $P(v_i, \mathbf{x}, t)$ to be a PDF, such that $P(v_i, \mathbf{x}, t)\, dv_i$ is the probability that the ith component of velocity is in the range dv_i about v_i at given values of $\mathbf{x}$ and t. There are also joint PDFs, for example, $P(\rho, v_i, \mathbf{x}, t)$, where $P(\rho, v_i)\, dv_i\, d\rho$ is the probability that the fluid variables have values in the range dv_i about v_i and $d\rho$ about ρ at given $\mathbf{x}$ and t.

The probability density function P is normalized so that

$$\int_{-\infty}^{\infty} P(v_i, \mathbf{x}, t)\, dv_i = 1. \tag{12–1.13}$$

The value of the turbulence-averaged variable v_i is obtained from

$$\int_{-\infty}^{\infty} v_i P(v_i, \mathbf{x}, t)\, dv_i = \overline{v}_i(\mathbf{x}, t). \tag{12–1.14}$$

The functional forms assumed for PDFs are obtained from measurements or other physical considerations that are outside the scope of the macroscopic equations. For example, a PDF can have delta functions at its extremities, representing the pure fluids that exist on either side of a convoluted interface, and a continuum portion that may be flat, Gaussian, or some other shape in between. It is possible to develop evolution equations for PDFs that look like conservation equations. As with stress-transport models, however, this approach requires additional phenomenological expressions to close the equations. There have been some solutions to compute the correct form of the PDFs for a single scalar by finite-difference methods, and more recently there have been Monte Carlo solutions deriving joint PDFs. A more formal approach that derives equations for the evolution of PDFs is described by O'Brien (1980). We recommend the two review articles by Pope (1985, 1994) for a comprehensive discussion of PDFs.

The PDF approach remedies some of the problems with turbulence approaches based on straightforward Favre averaging. One problem with PDF representations is the relative difficulty in determining mass-averaged quantities experimentally, which makes it difficult to calibrate the models. As with turbulence-averaged approaches, the use of a PDF implies some kind of average in time, in space, or over an ensemble of flows. Since space and time variations appear explicitly in equations (12–1.13) and (12–1.14), simple PDF approaches based on such implied averages share the same lingering inconsistency as turbulence-averaged approaches. We return to consider these methods further in Section 12–4 on turbulent reacting flows, an application for which PDFs have been used with notable success.

12–2. Large-Eddy Simulation and Subgrid Models

For turbulent flows spanning a wide range of space scales, in geometrically complex domains, or complicated by other physical processes such as chemical reactions or radiation effects, it is not practical to perform DNS on currently existing computers. It might not even be practical to compute such flows on computers that will exist twenty years from now. For example, a three-dimensional model resolving the full range of scales of the air flowing through a modern turbofan engine might need 10^6 to 10^7 computational cells in each spatial dimension, for a total of 10^{18} to 10^{21} grid points. It is possible to use 10^9 cells for a few special problems, but this is not nearly enough, by many orders of magnitude, to resolve the full dynamic range of the flow in the engine. Rather than throwing up our hands in amazement and perhaps finding other work to do, it is worth the effort to try other, less exact approaches that still allow useful information to be extracted.

Large-eddy simulation is also based on the idea of separation of scales and separates the flow into two regimes. The first regime, comprised of the large, macroscopic scales that can be resolved numerically, is simulated using models based on the Navier-Stokes or Euler equations and depends on the physical boundary and initial conditions. The second regime, containing the small scales, ranges from the grid-scale cutoff in the large-scale simulation down to l_K. The large scales are assumed to contain most of the turbulent energy. The physics on the smaller scale has to be represented by *subgrid turbulence models*. The small scales are coupled to the local macroscopic flows in a way very similar to that used for the one- and two-equation models. There is a fundamental difficulty, or perhaps inconsistency, in the inherent assumption of a separation of scales. This, however, may be mitigated for LES models when the large, resolved scales and the small, modeled scales are distinct and so do not overlap too much. As Figure 12.1 shows, the turbulent kinetic energy spectrum is continuous. Thus an important part of LES is guaranteeing that the large and small scales behave continuously and physically at the transition between the two representations.

Modeling subscale phenomena usually means representing physical or chemical properties that occur on length scales smaller than a computational cell. The cell size is a natural and convenient scale at which to do the matching since the larger scales can be resolved. Nonetheless, the grid discretization complicates the theoretical models, because grid truncation errors are unique to the particular solution algorithms used to simulate the large scales. Modeling subscale phenomena can also require representing certain sorts of flow physics in a direction that is not resolved numerically. Examples include turbulent convective transport in a second or third dimension of a one- or two-dimensional problem. Sometimes the definition of subgrid modeling extends to models that represent other physical processes that can alter the flow, such as droplet evaporation or particulate condensation. Generally, the term *subgrid modeling* refers to subgrid turbulence modeling. When LES subgrid models are used, there is still the fundamental problem of using averages of flow quantities at the same time those quantities are varying in space and time. Even though the errors that might now be incurred should be mostly confined to a range below the grid scale, they still could have some effects on the large-scale solution.

12–2.1. Overview of Large-Eddy Simulation

There are three basic parts of LES models:

1. Navier-Stokes (or Euler) "resolved-scale" simulations of the largest scales, sometimes filtered,
2. subgrid models of the effects of fluid flow occurring on space scales which are not resolved, and
3. methods for coupling the subgrid models to the resolved-scale simulations.

When a numerical solution of the Euler or Navier-Stokes equations resolves only the largest scales, there is a numerical cut-off determined by the computational cell size. This cut-off is considerably larger than the Kolmogorov scale for moderate– and

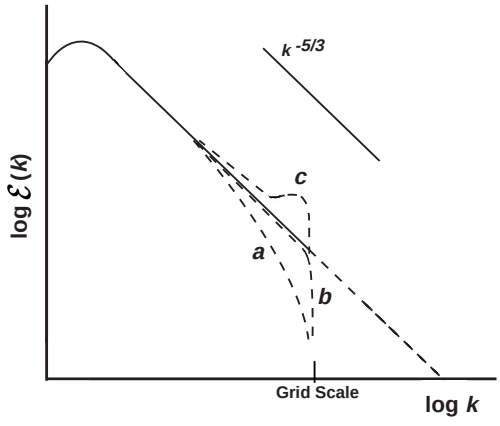

Figure 12.2. Kolmogorov spectrum showing schematically how different algorithms used for solving the Navier-Stokes or Euler equations behave differently at the grid-scale cut-off. (a) Algorithm is too diffusive. (b) MILES algorithm. (c) Effects of spectral blocking.

large–Reynolds-number flows. The different numerical algorithms for integrating convective flows (see Chapters 8 and 9) result in very different truncation errors in the solutions at the grid cut-off. This is illustrated schematically in Figure 12.2, which shows several different behaviors that could occur at the grid-scale cut-off in a numerical simulation. For example, if an algorithm allows the energy to become trapped and then accumulate at the grid scale without sufficient dissipation (see Figure 12.2, curve *c*), the method becomes unstable, or at least the results become unphysical as the large scales become contaminated. When turbulent energy is trapped at the grid scale rather than cascading down to the Kolmogorov scale, the energy-transfer rate out of the large scales might be too slow. Some algorithms dissipate turbulent energy at the small grid scales, so that the kinetic energy of the small-scale motions is converted into heat (curves *a* and *b*). Thus they mimic viscous dissipation but at the wrong scale. This numerical dissipation is dangerous when its effects extend too far up from the grid scale into the resolved scales (curve *a*).

One commonly used way to adjust the amount of energy extracted from the resolved scales is to apply a filter to the equations being solved. Such filters serve two purposes: they partially suppress the grid truncation errors, and they provide the basis of a formalism for estimating of subgrid eddy transport. Ideally, enough filtering will allow a smooth connection between the resolved and subgrid scales. A filtering procedure can also remove the oscillations caused in many convection algorithms by the Gibbs phenomenon. Filtering techniques are used with spectral algorithms and high-order compact finite-difference algorithms (see Chapter 8). When these linear, high-order, low-dissipation algorithms are used to simulate high–Reynolds-number flows, the physical viscosity may be too small to prevent unphysical energy build-up in the smallest resolved scales in the absence of a filter.

Thus conventional LES approaches do not solve the Navier-Stokes equations directly; they solve a derived set of equations for filtered variables,

$$\overline{f}(\mathbf{x}, t) = \int f(\mathbf{y}, t)\, G(\mathbf{x} - \mathbf{y})\, d\mathbf{y} = \int f(\mathbf{x} - \mathbf{y}, t)\, G(\mathbf{y})\, d\mathbf{y}. \tag{12–2.1}$$

Here f is a scalar or vector, such as ρ, $\rho\mathbf{v}$, or E, and the barred quantities represent the values for the larger resolved scales. The filter G is usually taken as a low-pass filter, often a Gaussian (see, e.g., the summary in Rogallo and Moin [1984]). Many filter functions $G(\mathbf{x} - \mathbf{y})$ have been tried, each shape having a corresponding averaging scale. There

should be, in principle, an optimum filter that minimizes the error in the calculation, and this would depend on the CFD algorithm used, the scale over which the filter acts relative to the smallest computed scale, and perhaps on the subgrid model that will be added to the solution.

The result of applying a spatial filter to the Navier-Stokes equations is a set of equations that resemble the Reynolds-averaged equations discussed in Section 12–1. Now the equations contain terms such as the subgrid-scale tensor,

$$\mathcal{T}_{ij} = \overline{v}_i \overline{v}_j - \overline{v_i v_j}, \tag{12–2.2}$$

which is different from the Reynolds stress term equation (12-1.6), and so must be modeled and interpreted differently. The most common assumption for $\mathcal{T}_{ij}$ is:

$$\mathcal{T}_{ij} = 2\nu_t \overline{S_{ij}} + \frac{1}{3}\mathcal{T}_{kk}\delta_{ij} \tag{12–2.3}$$

with

$$\overline{S_{ij}} = \frac{1}{2}\left(\frac{\partial \overline{v_i}}{\partial x_j} + \frac{\partial \overline{v_j}}{\partial x_i}\right). \tag{12–2.4}$$

This representation introduces the eddy viscosity ν_t. A repeated index indicates that a summation is taken over the index. A popular model for ν_t, proposed by Smagorinsky (1963), defines

$$\nu_t = (C_s\,\Delta x)^2 |\overline{S}|, \quad \text{and} \quad C_s \simeq 0.1 - 0.18. \tag{12–2.5}$$

Various models for ν_t and $\mathcal{T}_{ij}$ have been developed and studied extensively since the early work of Smagorinsky. Subsequently, more complex models for subgrid turbulence that consider the dynamic behavior of ν_t, $\mathcal{T}_{ij}$, C_s, and other parameters were (and still are being) developed. Rogallo and Moin (1984) and Lesieur and Métais (1996) provide excellent summaries of the hierarchy of these models, indicating where they excel and where they break down. A useful review, including comparisons of a wide range of subgrid models for LES, was given by Fureby (1996).

Renormalization group methods mentioned in Section 12–1 have been used to provide a form of ν_t suitable for LES (see Yakhot and Orszag [1986a,b]; reviewed by McComb [1985], Herring [1985], and more recently by Smith and Woodruff [1998]). These methods are based on a systematic, progressive elimination of the effects of the smallest scales, replacing their mean effect on the larger scales by an effective turbulent viscosity. RNG has been used to derive models for both RANS and LES. Lam (1992) has presented a useful analysis of RNG.

It is important to consider the ways in which modeling the small scales affects the computations of the large scales. One effect that is often left out of LES is *stochastic backscatter*. Turbulence on very small scales can feed some energy into large scales where it can grow by exciting various fluid instabilities (see, e.g., Leith [1990] and Piomelli et al. [1991]). Although these models generally assume that subgrid phase information is not important for large-scale dynamics, there are issues concerning short-wavelength motions with a long-wavelength component. Even though these small-scale motions may average to zero, they can have an important effect on long wavelengths. The small-scale

motions can act as a source of long-wavelength perturbations when the phases of widely separated small scales are uncorrelated (Fox and Lilly 1972; Herring 1973). Unlike the incoherent part of the spectrum, this is not a diffusion-like term. Rather, it appears as a "random" fluctuation in the large scales caused by the small scales. Fox and Lilly suggested introducing random fluctuations that react back onto the large scales. A computational experiment carried out by Leith (1990) began the task of examining the effects of stochastic backscatter on the large-scale flow.

Another interesting and related issue is what could be called an uncertainty principle for LES. As explained by Lesieur and Métais (1996), the LES problem is not well posed mathematically. If there is no initial information about the small scales, the uncertainty in small-scale motions will eventually contaminate the large scales. This error cascade corresponds to a decorrelation between two different realizations of the flow that differ initially only at the smallest scales. Whatever the precision of the numerical method used, the LES prediction must eventually decorrelate from the exact solution. This is probably no problem in practice, because the lack of determinism mirrors the physically correct sensitivity of a flow to minute differences in the physical initial conditions. The large-scale features and the statistics of the flow are insensitive to minute differences in initial conditions, and they are still correctly predicted. This fascinating point has been discussed by Lorentz (1969), Leith and Kraichnan (1972), and Herring (1973). We will see later that this insensitivity is extremely important to the successful modeling of turbulence. Efforts have been made to incorporate some of the effects of small scales on the resolved scales, including the systematic approach of RNG. Backscatter or uncertainty has implications for the accuracy of, for example, long-term weather forecasting.

12–2.2. The Ideal Subgrid Turbulence Model

Computational research on turbulent flows is often pursued as a two-step, bootstrap process. DNS is used to resolve the full range of physical scales, and is performed for systems in which the range of scales present is limited and can be resolved computationally. These DNS computations augment theory and experiment by providing information about the important nonlinear mechanisms in turbulence. This understanding, in turn, is used to calibrate the accuracy and performance of more economical and widely applicable LES simulations and other turbulence models.

We now summarize the material we have presented in the form of a list of properties that should be incorporated in an ideal subgrid turbulence model. The computational consequences of these properties are important because they can provide a guide for extracting information from DNS about the nature of turbulence. In turn, this information can be used to construct practical models to use in simulations of turbulent flows.

Sufficient Generality

The ideal turbulence model should be sufficiently general to represent fluid dynamics and other subgrid properties of the system. This means that it should handle properties such as compressibility, high–Mach-number flow, and multispecies effects, as required. The model should predict the physically correct subgrid-scale mixing on the correct macroscopic space and time scales.

Conservation

The model should satisfy the global conservation laws of the entire system, taking into account the resolved and unresolved scales. Conservation should be enforced in LES and turbulence models, both separately and taken together.

Minimal Contamination of the Resolved Scales

The turbulence model should ensure that the macroscopic, well-resolved scales are only minimally contaminated by whatever the model does at the small scales. Such contamination may arise by inaccurately resolving flow structures on the grid scale or by the particular properties of a numerical procedure, such as filtering the variables.

Smooth Connectivity

Breaking the calculated flow into large and subgrid scales is an artifice. The scales in turbulence are contiguous over the entire spectrum of wavenumbers from the largest scale down to the dissipative scale. Therefore, the turbulence model should smoothly connect to the predictions of the resolved scales at each point in space, even when the grid size varies. The effects of all scale lengths, whether modeled or resolved, should be included only once. The large-scale and subgrid-scale spectra must merge smoothly at k_{cell}, the wavenumber associated with the computational cells. If k_{cell} is varied by changing the numerical resolution, the predictions of the turbulence model coupled to the fluid equations should not change. Any subgrid or turbulence model that does not specifically take the grid-scale algorithm errors into account cannot perform this matching accurately.

Onset and Other Transient Laminar Phenomena

A turbulence model should not *a priori* assume the flow is turbulent. A fluid-dynamics simulation with a good turbulence model should be able to predict the onset of turbulence in an initially laminar flow. The density, temperature, and velocity gradients resolved in the simulation drive fluid-dynamic instabilities, which in turn eventually drive subgrid turbulence. The resolved, macroscopic flow properties determine the energy available to drive the subgrid turbulence, so mechanisms responsible for the onset of turbulence should not be masked by the turbulence models.

Coupling between Large-Scale and Subgrid-Scale Motions

The turbulence model is driven by information from the large-scale flow. In addition, the effects of the unresolved, modeled turbulence enter the macroscopic flow equations through phenomenological terms representing averages over the subgrid dynamics. Examples of these terms are eddy-viscosity coefficients, diffusivity coefficients, and average chemical heat-release terms that appear as sources in the macroscopic flow equations.

Because subgrid fluctuations can act as infinitesimal triggers of instability, a source term introducing small, random fluctuations is required wherever there are any significant subgrid motions. This source term provides a feedback mechanism to describe stochastic energy transfer from the small scales, where mixing and subsequent chemical reactions occur, to the large-scale flow. The term appears in the equations in addition to the averaged diffusionlike effects.

Complicated Reactions and Flows

It seems obvious that a model cannot account for physical effects based on processes not included in the model, but sometimes the situation can be more subtle. No matter how general a turbulence model is, it cannot be expected to do everything. A fluid-dynamic turbulence model alone cannot be expected to account for the effects of chemical kinetics, buoyancy, droplets, condensation, and other processes that result from additional physics not in the basic Navier-Stokes equations. Because subgrid models are needed to represent many different kinds of physical processes, the framework adopted should be general enough to allow these to be added, as needed, at both the large and small scales.

For example, if there is an appreciable time delay as vorticity cascades through the inertial range to the short-wavelength end of the spectrum, the model should be capable of representing this delay. Such a capability would allow the description of bursts and other intermittent phenomena that trigger chemical reactions to have the correct timing. Therefore, reactive-flow subgrid models will have to be added to the fluid dynamics, not treated entirely separately.

Economy

A nearly infinite number of subgrid degrees of freedom must be represented by a very finite number of variables. Therefore, any practical subgrid model is intrinsically incomplete and must trade off accuracy with economy. For the model to be generally usable, it is important to keep the number of subgrid variables in each computational cell to a minimum.

12–2.3. Four Fortunate Circumstances for Turbulence Simulations

Over the last twenty five years, we have observed that multidimensional, time-dependent simulations, performed using monotone fluid-dynamic algorithms, produce results that agree well both qualitatively and quantitatively with experiments on transitional and turbulent systems. These results have been obtained even though no explicit turbulence model was included in the computations, although there was good reason to believe that there was significant turbulence on the unresolved scales.

The process of discovering and assembling the physical and numerical arguments to explain this puzzle illustrates the inseparable interplay between advances in our knowledge about turbulence and improvements in numerical simulation capabilities. This effort has led us to note, with some relief, that there appear to be several related circumstances that conspire to make simulating practical turbulent flows possible on existing computers. Because the fluid has the physical behavior it does, a great simplification for modeling the turbulence results.

The Shape of the Kolmogorov Spectrum

The first fortunate circumstance is that the spectrum of the turbulent energy flow, the Kolmogorov spectrum (see Figure 12.1) falls off at small scales at a very convenient rate. *The average kinetic energy, associated with progressively smaller turbulent scales, decreases fast enough for the scales containing most of the energy to be resolvable by current three-dimensional simulations.*

If the spectrum were somewhat flatter, for example, if it scaled as $(kL)^{-1}$, most of the energy would be in the small scales. It would then be necessary to resolve these small scales to capture most of the turbulent energy. On the other hand, the Kolmogorov spectrum is also not too steep. The velocity of the small eddies is still large enough that their rotational turnover time increases rapidly with decreasing eddy size. Though the small eddies are relatively insignificant energetically, they still turn over considerably faster than the larger eddies. *This means that there is enough energy for the small scales to mix large-scale inhomogeneities as fast as the large-scale flows can produce them.* Various shapes of spectra are shown schematically in Figure 12.3.

An example of the mixing that arises in a computation can be seen in Figure 12.4. Small, secondary instabilities grow, saturate, and mix fluid during the linear growth phase of the primary Kelvin-Helmholtz instability of an elliptical jet. The details of the molecular mixing are relatively unimportant on the scale of the macroscopic motions as long as the intermediate and small scales of motion guarantee mixing. If the Kolmogorov spectrum were much steeper, unresolved molecular mixing would not necessarily occur quickly inside a turbulent large-scale eddy.

Energy Transfer through Local Interactions

The second fortunate circumstance is supported by both theory and simulations using DNS: *Turbulent energy is transferred by a turbulent cascade that passes through the intervening, nearby, or local scales to the small scales where it is eventually dissipated.* DNS of homogeneous turbulence was performed specifically to study the nature of the coupling between the large and small scales (Domaradzki et al. 1987). The simulations showed

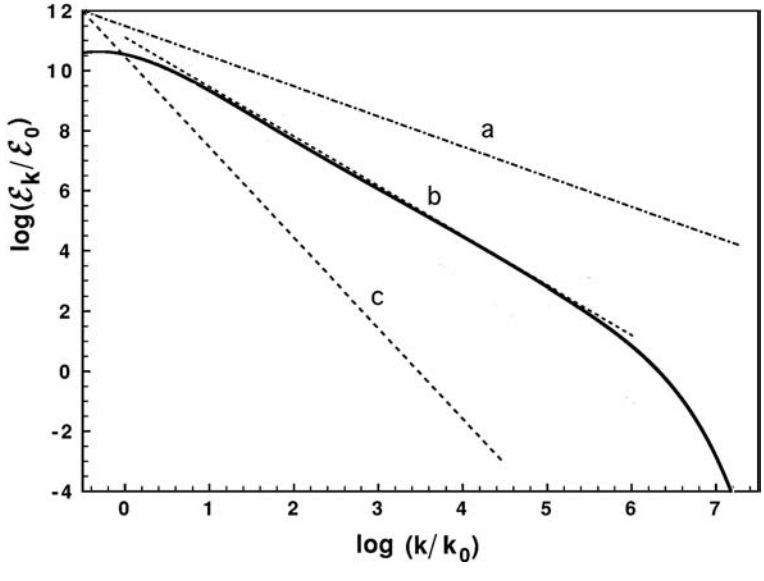

Figure 12.3. Possible spectra of the decay of the turbulent kinetic energy $\mathcal{E}$ as a function of wavenumber k. (a) Spectrum scales as k^{-1}, which is too flat. Most of the energy is in the small scales. (b) The $k^{-5/3}$ spectrum for three-dimensional turbulence. (c) Spectrum too steep, so that small scales dissipate the energy too fast.

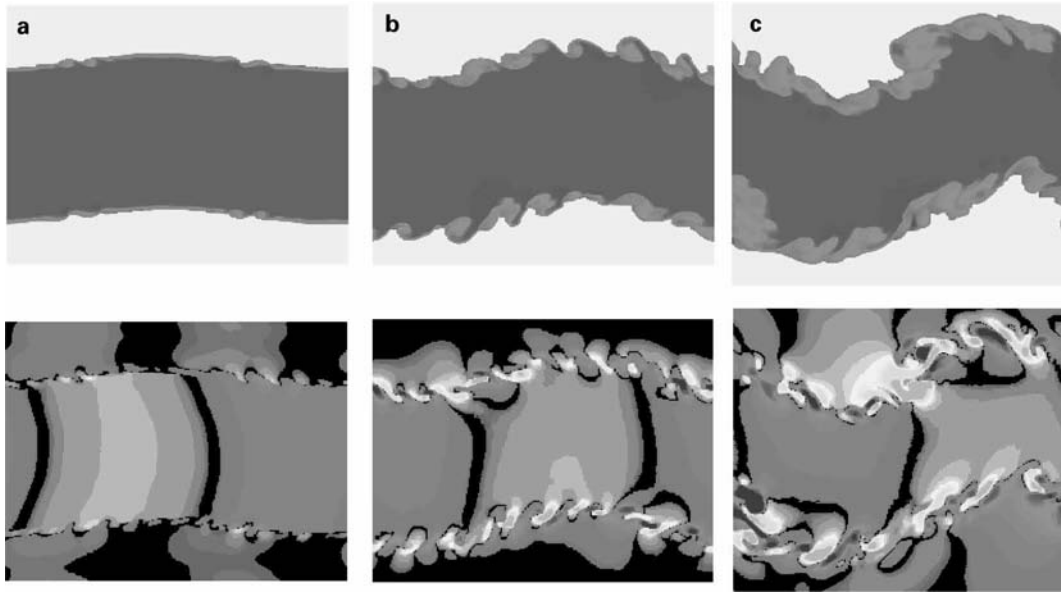

Figure 12.4. Cross-sectional views reveal the mixing of fluid in an unstable, elliptical jet flowing from left to right at 100 m/s (Boris et al. 1992a). The views are through the center of the jet along its major axis. The simulations were done on a 256^3 grid using an FCT algorithm. Upper panels: lightest gray indicates ambient air; dark indicates unmixed jet gas; intermediate shades indicate the mixed-fluid region. Lower panels, vertical velocity: black indicates small vertical velocity. (a) 0.4 ms, (b) 0.9 ms, and (c) 1.35 ms.

that *energy transfer in the inertial range is dominated by local interactions* (Yeung and Brasseur 1991; Domaradzki 1992). In practice, virtually all of the energy extracted from a given scale occurs as a result of interactions with eddies at most an order of magnitude smaller. This is in contrast to the possibility that the energy might be deposited directly from the large, energy-containing scales into the small scales. If the very smallest scales were dynamically significant and extracted a significant amount of energy directly from the large scales, there would be no good alternative to resolving all the scales. Instead, the actual, predominantly local cascade of energy guarantees that numerical simulations that resolve the bulk of the energy-containing scales can give a reasonably accurate representation of the energy transfer out of those scales.

This relatively smooth transfer also means that there is a large "inertial" region in the spectrum where the behavior of the fluid dynamics is essentially scale invariant. This region of a straight-line spectrum, with the $-5/3$ slope, becomes an acceptable place to match a subgrid-scale model to the resolved-scale model.

Dynamics on the Large Scales

The third fortunate circumstance is *the apparent lack of important dynamics occurring at scales even a factor of ten or more larger than the classically defined Kolmogorov scale.* The dissipation of structures in the flow at scales appreciably larger than the Kolmogorov scale is sufficiently strong that little structure actually survives to reach the small scales. This further reduces the need for spatial resolution. Evidence of this comes from experiment, theory, and simulations (Moin and Mahesh 1998; Vuillermoz and Oran 1995).

Behavior of Monotone Algorithms at the Grid-Scale Cut-off

The fourth fortunate circumstance is really a property of the class of numerical convection algorithms called *nonlinear monotone methods* rather than an observation about turbulence. These algorithms are based on principles of conservation, causality, and positivity (more generally, monotonicity), as described in Chapter 8. They have the property that the local nonlinear dissipation in the algorithm smoothly connects the large, resolved, energy-containing scales to the unresolved subgrid scales and provides a built-in measure of the dissipation required. If we use these methods for DNS, we can believe the results of computations resolving eddies only down to l_K or even a factor of ten above it. If we use these methods for LES, we can rely on their computing the large scales correctly without significant contamination from numerical errors at the smallest scales resolved.

12–3. Turbulent Reactive Flows

How much of the information presented in this chapter for nonreacting flows can be carried over to computations of turbulent reacting flows? Specifically, which ideas carry over, and which methods can be used? How does the presence of a reacting material change the nature of turbulence? As we indicate in the following discussion, many of the major ideas carry over directly. In fact, if there is no energy release due to reactions, so that the coupling with the fluid dynamics is relatively weak, then the concepts and methods discussed earlier carry over directly. DNS, RANS, and LES, as well as the MILES concepts presented in Section 12–4, can be applied with minor modifications to account for the presence of a number of species and more complex mixing processes. This is the situation, for example, in many atmospheric flows where the reactions are essentially thermoneutral and the heat input is from an external source and relatively slow. Serious differences can occur, however, when energy changes control the flow, particularly the type of fast, intense energy release that occurs in combustion. And most real combustion flows are turbulent. Combustion in engines, furnaces, and even in thermonuclear supernovae is turbulent. For this reason, it is extremely important to have computational models that can compute flows in turbulent combustion regimes (see Liñán and Williams [1993]).

12–3.1. Some Problems in Modeling Turbulent Reactive Flows

Here we describe some of the physics of reactive turbulence. In particular, we emphasize the extreme case in which turbulence with strongly exothermic chemical reactions differs from nonreactive turbulence. Local energy release causes a highly nonlinear feedback process between the reacting species and the fluid dynamics. The effects are compounded by the typically exponential form for the reaction rates and energy release. For example, a change in temperature can raise the viscosity, which then lowers the Reynolds number, and changes the dissipation scale. This implies that we might need less resolution behind a turbulent flame front, where the Kolmogorov scale is relatively large, than in front of it, where the flow is cold! The energy release in a turbulent flame also causes the fluid to expand, which can generate appreciable vorticity and trigger fluid instabilities that further complicate the flow. Turbulence alters mixing and reaction times and heat and mass transfer rates, that, in turn, modify the local and global dynamic properties of the

system. The following discussion extends and expands the consideration of fluid dynamics and chemical reactions in Section 2–2.6.

Dynamic Fluid Instabilities and Turbulence Generation

A number of flow instabilities can contribute to the onset of turbulence. At each scale, the quick onset of short-wavelength instabilities results from variations of the background flow at that scale. When the background flow variables are near the instability threshold, the growth rates are very slow and the unstable wavelengths relatively long. As the background flow evolves into an unstable regime, the maximum growth rates of the most unstable modes increase, and instability spreads over a broader band of the spectrum. The turbulent generation of increasingly disordered vorticity is the result of the nonlinear evolution of these instabilities and is an intrinsically transient phenomenon.

Two instabilities commonly discussed in the evolution of turbulence in variable density flows are the *Rayleigh-Taylor* and the *Kelvin-Helmholtz* instabilities. The Rayleigh-Taylor instability is caused by the acceleration of a heavy fluid through a light fluid. In this instability, vortex pairs of opposite sign are generated, for example, in buoyant flows or when sound or shock waves interact with flames or density gradients. Rotationally generated Taylor instabilities also have the same character. The Kelvin-Helmholtz instability occurs at the interface between flows of different velocities. It is responsible for the initial formation of coherent structures in splitter-plate and jet experiments. It also tends to fragment existing sheets of vorticity into roughly parallel vortex filaments of the same sign. The existence of strong density gradients is one important feature of reactive flows that is not generally considered in classical nonreactive turbulence models.

Density gradients generate vorticity through interaction with ambient accelerations, often apprearing as gradients in the pressure field. This is the $\nabla\rho \times \nabla P$ term in the vorticity evolution equation (2–2.8). When the mass density is constant, all of the effects resulting from this vorticity source term are absent. The Rayleigh-Taylor instability also leads to strong effects when density gradients interact with an external pressure gradient, such as would be caused by gravity.

The various types of instabilities that can arise in combustion systems are extensive, including chemical-acoustic, thermo-diffusive, thermal, and Landau-Darrieus instabilities. All of these can produce perturbations that increase surface areas and mixing. These instabilities and their effects on reacting flows are discussed in many texts (see, e.g., general combustion books, such as the texts by Williams [1985a] and Zeldovich et al. [1985]).

In combustion, the localized heat release can cause strong, transient expansion of the reacting gases. The resulting low-density region has gradients with scales characteristic of the combustion process. When these density gradients interact with existing pressure gradients, they *passively* generate vorticity on these same characteristic scales. Vorticity on these scales is particularly efficient at mixing. The combustion process is, in turn, enhanced as long as fuel and oxidizer are present. Because the turbulent spectrum now seems to be driven energetically at short wavelengths as well as long, plateaus or even peaks might form in the turbulence spectrum, and these can change the usual notions of cascade and scaling.

Heat released in an exothermic process can also have an active effect on turbulence. The time-varying acceleration accompanying the expansion causes a transient acceleration

in the surrounding fluid. This active role can be understood qualitatively in terms of the effect in a fluid of a Rayleigh-Taylor instability, which occurs when the expansion causes an acceleration to act on a density inhomogeneity or gradient. When the scale of a density gradient is comparable to the distance the fluid moves while expanding, the perturbation in the density gradient grows by a factor of three (exponentiates once) during the expansion, leaving some residual vorticity in the flow. The vorticity from this transient expansion has a characteristic velocity of only a few percent of the velocities in the driving expansion (Picone and Boris 1983), but it persists for a long time. Because of the importance of the expansion process, models for nonreactive turbulence, designed to account for Rayleigh-Taylor instabilities, would also apply to the reactive models with heat release.

The *passive role* of expansion is potentially more important than the active role. In the passive role, expansion influences turbulent mixing by providing density gradients, rather than the pressure gradients, that subsequently lead to vorticity generation. Because the active generation of vorticity is limited in duration to the time of the expansion phase, it is reasonable to expect that the amount of vorticity generated by the passive interaction of density gradients with large-scale vortices is greater. There are several growth times of the Rayleigh-Taylor instability during a single rotation of the large vortex when the mode wavelength is smaller than the vortex radius.

Expansion caused by energy release also plays an important *local role* in the interaction between chemical kinetics and fluid dynamics. Density, temperature, and composition changes that occur because of chemical reactions, can modify the fluid dynamics through diffusive transport processes and changes in the equation of state.

Reactive Interface Dynamics

In turbulent systems where the reacting species are not premixed, the microscopic, molecular mixing depends on the motion and contortion of Lagrangian interfaces that originally separate reactive species. We must understand and be able to predict the behavior of these surfaces as they move and stretch with the fluid. As the fuel and oxidizer diffuse into each other at the molecular level, these surfaces continue to lie roughly normal to the strongest species density gradients. Rapid increase of the reactive surface area due to turbulence enhances molecular mixing and usually speeds reactions. The situation is shown schematically in Figure 12.5. In the upper left side of the figure, a cube of fluid 1 cm on a side is shown at time $t = 0$. An interface between reacting species A and B divides the cube in half. Subsequent convolution and stretching of this reactive surface is controlled by the vorticity distribution in the fluid.

Motion of the fluid at the reactive surface breaks down into a velocity component normal to the surface that can change the total amount of surface area, and components parallel to the surface that cannot. Both of these two components, shown in Figure 12.6, are important and complement each other in the turbulent mixing process. The component normal to the surface causes interpenetration of parcels of fluid from both sides of the interface, a process called *entrainment*. When nearby points on the surface separate so that the surface stretches, the fluids on opposite sides of the surface approach the surface, and thus keep the flow roughly divergence free. Species and temperature gradients normal to the surface are enhanced. This, in turn, also increases the diffusive interpenetration

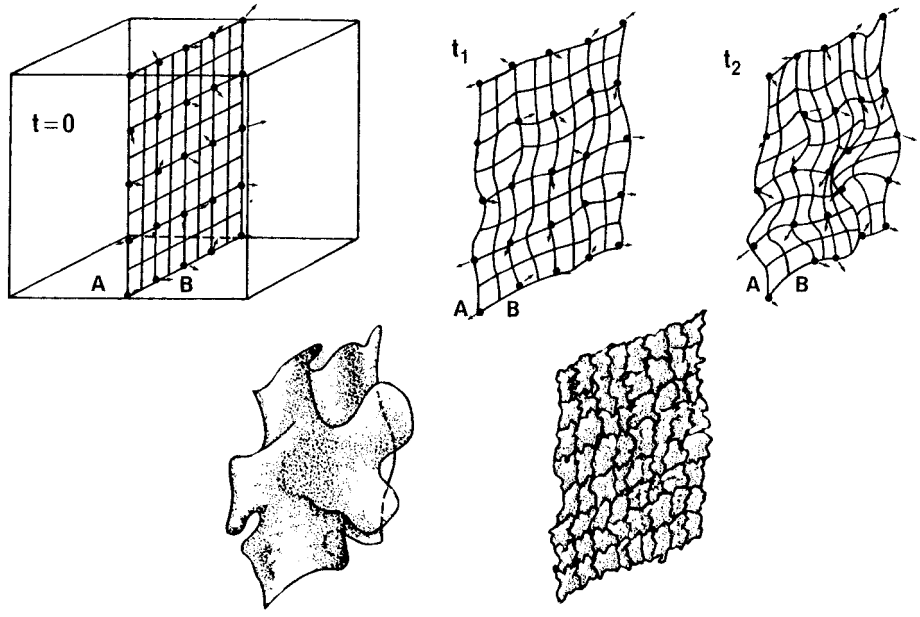

Figure 12.5. Top panels: Three stages during the early deformation of a reactive interface between two fluids A and B. At $t = 0$, the surface separating A and B is shown as flat in a three-dimensional volume of fluid. At later times t_1 and t_2, the surface progressively deforms as the result of the Lagrangian motion of surface points. The small arrows indicate the local direction of flow. Lower panels: Two different cases of increase in the area of a reactive interface. When molecular diffusion is important and the turbulence velocity spectrum is enhanced at short wavelengths, the surface is a relatively flat, fluffy surface. When diffusion is small and the velocity spectrum is large at longer wavelengths, and smoother bulges result.

of the reactants. This surface-stretching process is independent of material entrainment. By increasing the actual area of the reactive surface, the bulk reactivity is also increased. Stretching and interpenetration usually occur simultaneously, but for simplicity are shown separately here.

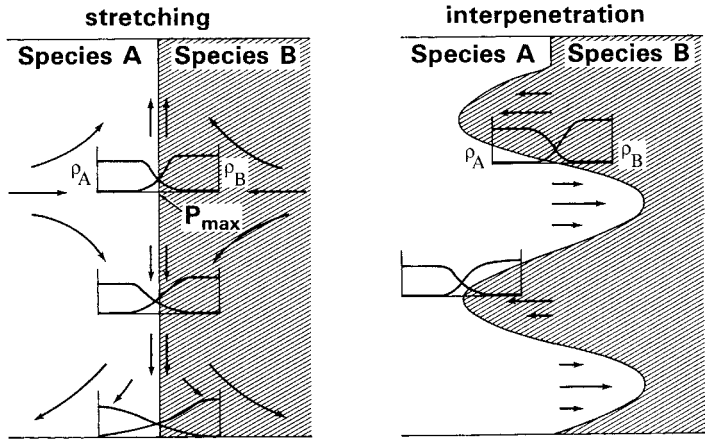

Figure 12.6. Interpenetration of two species A and B as influenced by local dynamics at the reactive interface.

When the rate of molecular diffusion is slow, and diffusive mixing is restricted to very small scales, as in some liquids, there can be significant entrainment without molecular-scale mixing. This situation maintains distinct reactive interfaces (and is probably best treated with PDF methods considered again later). Nonetheless, many aspects of turbulent combustion can be considered as an extension of nonreacting, gas-phase turbulence if the small scales of motion are sufficiently populated so that mixing on those scales occurs fast compared to motion on the large scales.

Detailed Chemical Kinetics

The energy released by the chemical reactions is a complicating aspect of reactive-flow mixing. It affects the fluid dynamics by causing expansion, and it changes the physical conditions that affect reaction rates. In some cases, the overall reaction rate is governed first by the diffusion of hot fuel and oxidizer together through an expanding region of hot products and reactants, and then by the subsequent reaction rates. In other cases, the fuel and oxidizer have time to mix thoroughly before chemical reactions occur. In these latter cases, the mixture can be ignited by a rapidly moving flame that travels parallel, rather than perpendicular, to the reactive surface. The situation is complicated significantly by flame stretching, which can extinguish reactions long enough for reactants and hot products to mix. In all cases, the timescales of the reactions have an important role to play in determining the type and speed of the reaction waves and the interaction with the fluid dynamics.

12–3.2. Overview of Turbulent Combustion

There are a number of excellent books and review articles that describe current concepts in turbulent-combustion and provide overviews of the latest approaches for solving a turbulent-combustion problem. Bray (1996) and the book *Turbulent Reacting Flows*, edited by Libby and Williams (1994) present excellent and comprehensive overviews of the physical problem and different methods of solution. We also recommend the reviews by Jones (1994), Pope (1990), Peters (1986), Bray and Peters (1994), Poinsot, Candel, and Trouvé (1996), and Vervisch and Poinsot (1998). Before describing some of the major approaches, it is necessary to describe several concepts used for discussing turbulent combustion.

A premixed reacting flow and a diffusional reacting flow are the two limits of turbulent reacting-flow problems. In fact, most problems lie somewhere between these limits and have some characteristics of both. These two limits are useful because they allow us to separate and study different effects. Premixed flames propagate through the material primarily as a reaction front that separates reacted from unreacted material. Diffusion flames require that significant mixing occur before reactions can even take place. This mixing usually occurs through diffusion at interfaces in a shear flow. For example, combustion in spark-ignition engines and thermonuclear supernovae is premixed, whereas combustion in furnaces, diesel engines, and gas-turbine combustion chambers require some mixing before reactions can occur.

There are a number of concepts and dimensionless numbers that are used in turbulent combustion, and these are helpful for defining different regimes of turbulent combustion. Above we defined $\mathcal{E}(k)$ as the kinetic energy associated with the turbulence scale k. Now

consider two sets of scales, one for large-scale turbulence in the system and one for the dissipative Kolmogorov scale. For the large scales, define t_{ls} as the characteristic time of the large-scale turbulence in the system, l_{ls} as the characteristic turbulent length scale, and Re_{ls} as the turbulent Reynolds number (often called Re_T in the literature). Then

$$t_{ls} = \frac{E}{\overline{\mathcal{E}}}, \qquad l_{ls} = \frac{E^{3/2}}{\overline{\mathcal{E}}}, \qquad Re_{ls} = \frac{E^{1/2} l_{ls}}{\mu} = \frac{E^2}{\overline{\mathcal{E}}\mu}. \tag{12–3.1}$$

Here $E = \int_0^\infty \mathcal{E}(k)dk$, $\overline{\mathcal{E}}$ is the rate of energy dissipation, defined as the rate of change of E due to viscous dissipation, and μ is the kinematic viscosity (see Table 2.4). Characteristic times and lengths for the small scales, now based on the Kolmogorov dissipation scale, are designated by the subscript K. If, for the smallest dissipative scales, t_K is the characteristic time scale and l_K is the characteristic length scale,

$$t_K = \left(\frac{\mu}{\overline{\mathcal{E}}}\right)^{1/2} \qquad l_K = \left(\frac{\mu^3}{\overline{\mathcal{E}}}\right)^{1/4} \tag{12–3.2}$$

and it can be shown that

$$\frac{l_{ls}}{l_K} = (Re_{ls})^{3/4}, \tag{12–3.3}$$

which becomes large at high turbulent Reynolds numbers.

It is also useful to define quantities for premixed turbulent flames, which are usually characterized in terms of the quantities defined above and three others. There are the flame time and length scales, t_f and l_f, and the laminar burning velocity, $v_f = l_f/t_f$, which is the velocity at which an idealized one-dimensional flame moves relative to the background gas. Then Da_T, the turbulent Damköhler number, and Ka_T, the turbulent Karlovitz number, are

$$Da_T = \frac{t_{ls}}{t_f} = \frac{E\, v_f}{\overline{\mathcal{E}}\, l_f} \tag{12–3.4}$$

$$Ka_T = \frac{t_f}{t_K} = \frac{Re_{ls}^{1/2}}{Da_T}. \tag{12–3.5}$$

From these definitions, four regimes of premixed turbulent combustion arise.

1. laminar flow, $Re_T < 1$
2. laminar flamelet combustion, $Ka_T < 1$, $Re_T > 1$
3. distributed combustion, $Ka_T > 1$, $Re_T > 1$, $Da_T > 1$
4. well-stirred-reactor combustion, $Re_T > 1$, $Da_T < 1$

In the laminar flamelet regime, burning occurs in thin, laminar flames that may be stretched or curved by the motions of the fluid. Because chemical reactions are fast and strong compared to the turbulence, it is hard to extinguish the flame. In the distributed combustion regime, the characteristic flame scales are intermediate between the Kolmogorov and large scales, and, in some regions of the flow, the flame may be quenched by the turbulence. In the well-stirred-reactor regime, the chemical time scales are large compared to turbulent time scales. These different regimes are best described by different turbulent models.

An important and natural question is whether we can make a similar categorization for diffusion flames. This is complicated by the observation that laminar diffusion flames, unlike laminar premixed flames, do not have a unique flame thickness or flame speed. What parameters, then, can be used to characterize diffusion flames? Some maintain that we should use the stoichiometric premixed laminar flame to provide the appropriate chemical scales. Then the concepts of the regimes described earlier would be applicable (see, e.g., the descriptions in Bray and Peters [1994] and Bray [1996]). This has led to the idea of defining a *mixture fraction space* $Z(\mathbf{x}, t)$ in which the region of the flame is viewed according to its relation to the stoichiometry of a premixed flame. The type of diffusion flame that can exist in a region of space depends on Z, and can be related to the four types of premixed flames listed above.

12–3.3. DNS and LES of Turbulent Reactive Flows

Direct numerical simulation of a reactive flow requires a very highly resolved solution of the reactive-flow conservation equations given in Chapter 2. The program that solves this problem may be constructed by selecting and combining algorithms from those described in Chapters 4 through 10, and using coupling methods described in Chapter 11. Then, given appropriate initial and boundary conditions, the model is solved on a three-dimensional grid that provides resolution down to the appropriate dissipative scale.

While computing nonreactive DNS of turbulence is expensive, DNS of a reactive flow is so much more expensive that it is prohibitive. If the problem involves a relatively simple geometry with only a few species whose chemical reactions do not cause significant energy changes in the flow, the simulation might be within the realm of what can now be done. For a flow as complex as a compressible, exothermic, multispecies flow, some modeling simplifications are usually required. Nonetheless, if properly constructed, such computations can provide extremely valuable information that is crucial for testing and clarifying the fundamental issues being studied. Such simulations can also be used to evaluate parameters for turbulence phenomenologies, such as those needed for RANS or PDF models.

To allow reasonably detailed computations to be performed in a practical way, simplifications are often applied to the chemical and diffusive-transport models. These simplifications are often justifiable based on a fundamental lack of information about the details of the chemical reactions. To be a DNS, the computations should still ensure that, within the framework of the actual problem being solved, all of the relevant physical scales down to the relevant dissipative scale are resolved. Reducing the chemical complexity in this way often produces results that give even more insight than solving the full, complicated system.

Another issue in DNS of reactive flows is determining *which* dissipative scales need to be resolved. If the flow is homogeneous and the turbulence is in equilibrium throughout the volume, it is possible to define a unique Kolmogorov dissipative scale. If the flow is dynamic, compressible, and there is local energy release resulting from fronts dividing burned and unburned material, there is no globally unique dissipative scale. Even if such a scale were associated with each fluid element, this scale would change dynamically as properties such as density, pressure, temperature, and constituent species change in time. The dissipation scale in the cold, unreacted flow may be orders of magnitude smaller than in a hot reacted flow, where the viscosity increases significantly with temperature. The cost

of the computation is significantly different depending on which scale must be resolved and the numerical resolution may have to change in time and space.

It is certainly possible to apply the methods of adaptive mesh refinement (AMR) discussed in Chapter 6 to turbulent reacting flows. We can, in principle, put the amount of resolution we think is needed when and where we want it. Further, consider a flow in which a high-speed, fairly fast flame is propagating through an energetic mixture. There are at least three dissipative scales to consider. One of these is the smallest scale, defined by the cold, unreactive, but potentially turbulent fluid. At the opposite extreme is the largest scale, defined by the properties of the very hot, turbulent, burned fluid.

Another important scale is the one required to resolve the flame itself. This usually depends on the complexity of the chemical-reaction mechanism. For a single-step Arrhenius model, ten or fifteen cells should be more than sufficient for most purposes. This might not suffice for a more complicated multispecies chemical model for the same physical system. Then, thirty, forty, or even more cells might be required to resolve all of the spatially and temporally varying chemical reactions and precursors that are required to compute the flame properties. The latter situation occurs, for example, in problems involving pollutant formation where species such as NOx, which are minor chemically but physically important, must be computed accurately.

We need to resolve the flame properly to compute its macroscopic properties. We also need to resolve the dissipative scale in the hot, turbulent flow to find the right level of perturbations in the cold flow. We probably can be more cavalier with the cold flow, since it is consumed anyway, and the MILES approach (described in Section 12–4) would give an adequate solution. But is this still DNS? It could probably give the same answers and have the same degree of usefulness as resolving the smallest scale required everywhere in the computation, but it would be much less expensive.

Recently there has been a rapid increase in the number of computations that can legitimately be called DNS of turbulent reacting flows. Most consider simplified chemical models, but there are some with much more complete, detailed models in simplified flow situations. Early work in this area used spectral methods (see Chapter 8) to simulate reacting shear flows with either no or relatively slow energy-release rates, very small density changes, and in idealized flow configurations (see, e.g., Riley, Metcalfe, and Orszag [1986]; McMurtry et al. [1986]; and McMurtry and Givi [1991], and references therein). Later work has generally used finite-volume methods in rather simple geometries, but for more complicated flows and with more complex chemical-reaction mechanisms. A recent review of the work performed in affiliation with the consortium that forms the Centre de Recherche sur la Combustion Turbulente (CRCT) is given by the series of articles in the book by Baritaud, Poinsot, and Baum (1996). In addition, there are comprehensive review articles on DNS in premixed combustion (Poinsot et al. 1996) and nonpremixed turbulent flames (Vervisch and Poinsot 1998). The relation of DNS to turbulence generally is discussed throughout many books and articles on turbulent reacting flows (see, e.g., Libby and Williams [1994] and Bray [1996]).

LES for Reacting Flows

In turbulent reacting flows, subgrid phenomena are faster, if anything, than in the corresponding nonreacting case. Thus, mixing delays are smaller and unresolved stratification of

distinct fluids should be important in less of the volume. In fact, because of the temperature-dependence of the viscosity in a gas, and the effects of a flame in effectively dissipating small scales as it burns through a gas, it is still correct that the viscous dissipation occurs at scales that are at least factors of five to ten times larger than the Kolmogorov scale. (This idea is consistent with the recent experimental results of Furukawa, Okamoto, and Hirano [1996].) Fast mixing means that chemical-reaction models assuming approximate spatial uniformity within a cell will often be adequate.

When the small-scale structure is not resolved, a subgrid-scale model might be necessary to describe the result of small-scale coupled reaction and diffusion processes. To date, LES and turbulence modeling for turbulent reacting flows have followed much the same approaches as used for nonreactive flows. Development of subgrid turbulence models has been an active area of research, and some of these models are described in Section 12–3.4. There is hope that relatively simple subgrid models can lead to reasonable predictions, and there may be no need for models that are as complex as those used in other approaches.

12–3.4. Turbulent Combustion Models

Many of the models that include the effects of turbulent combustion in simulations are extensions of the models originally proposed for constant-density flows. They are later extended to variable-density flows by using techniques such as Favre averaging. These models include $k - \epsilon$ models, various moment methods used in RANS, subgrid models for LES, and methods using joint probability distribution functions. It is important to note, however, that models whose origin is in the averaged equations of nonreactive turbulent flow do not take account of the length and the time scales of flame instabilities. Therefore, some care has to be taken when applying them. Their range of validity might be restricted to conditions in which the turbulent motion is sufficiently intense to overwhelm any velocity fluctuations associated with laminar flame instabilities. Unfortunately, many important combustion problems, as well as turbulent combustion data presented in the literature, are not in this regime. Reviews of second-order closure models applied to RANS and LES for turbulent reactive flows are given by Jones (1994) and Bray (1996).

There are, however, concepts and models particular to turbulent combustion that take into account some of the unique physical and chemical processes and the effects that occur in exothermic reacting flows. These include the Bray-Moss-Libby (BML) model and flamelet concepts summarized later. One of the important results that turbulent-combustion models can provide is the form of the mean reaction rate that has to be used in the macroscopic equations. For example, in some situations, there are significant fluctuations at small, unresolved scales. In such cases, and because of the exponential form, large errors are possible if a mean temperature is naively used to evaluate the reaction rate. Various models have been developed to deal with these situations. In all cases, however, these are phenomenological and their accuracy relies on how well they are calibrated.

Joint PDF Models

The joint PDF (JPDF) approaches are stochastic methods that provide subgrid information about the small-scale mixing and reactions. JPDFs must then be combined with traditional

RANS or LES methods to provide the cell-averaged, resolved-scale quantities needed as local source terms in a complete computation. Part of the appeal of JPDFs is that they are well-grounded physically. It has been shown that the exact equation for the JPDF can be derived from the Navier-Stokes equations (Pope 1985; Kollman 1990). Comprehensive reviews are given by Pope (1990) and Dopazo (1994).

Following the description in Section 12–1.5, the first step is to find the appropriate JPDF $P(\rho, T, Y_i; \mathbf{x}, t)$ and then to find the mean reaction rate $\overline{\dot{\omega}_i}$ by integrating $\dot{\omega}_i P$ over all the arguments, such as ρ, T, Y_i, and even velocity components. The discussion given below is cursory and illustrative. For example, suppose that there is just one chemical species and one scalar ϕ in the JPDF, where ϕ could be a physical property such as density or temperature. In addition, assume that P is really the Favre-averaged JPDF, so that P is a density-weighted quantity (defined in Section 12–1.3). The equation for P then takes the form (Bray 1996)

$$\overline{\rho}\,\frac{\partial P}{\partial t} + \overline{\rho} u_k \frac{\partial P}{\partial x_k} = -\frac{\partial}{\partial x_k}(\overline{\rho} < u_k'' > P) - \overline{\rho}\frac{\partial}{\partial \phi}(\dot{\omega}_i(\phi)\, P) - \frac{1}{2}\,\overline{\rho}\,\frac{\partial^2}{\partial \phi^2}(< \overline{\varepsilon}_{i,j} > \; P). \tag{12–3.6}$$

Here the average $<>$ indicates a mean (usually weighted) of the quantity, and $\overline{\varepsilon}$ is the rate of dissipation of turbulent kinetic energy. The equation has to be closed by modeling the terms $< u_k'' > \; P$ and $< \overline{\varepsilon}_{i,j} >$. Then it must be solved in conjunction with some model that provides mean quantities, here $\overline{\rho}$. It can then be used to evaluate the mean reaction rate.

Another approach, the presumed PDF approach (see, e.g., Borghi [1988] or Bray [1996]) replaces a PDF by an assumed form and so bypasses the solution of equation (12–3.6) The form could be a beta function, a clipped Gaussian, or some simple shape. The assumed form can also be a complicated or even conditional function of local system properties. It is used to determine the mean reaction rate. These methods work best when the number of independent variables is small. When there are many variables, it is useful to find a form of the PDF that excludes unphysical regimes imposed by the thermodynamics or reaction processes. The simpler, empirical approach is appealing, but not completely straightforward to implement.

The PDF methods are stochastic approaches for describing small-scale mixing. They are best used for the situation in which $Da_T \ll 1$, that is, for cases where the chemical times scales are long compared to other characteristic timescales in the system. The opposite limit is the case of flamelets, for which the chemical reaction times are relatively short. Starting with this assumption gives a rather different set of subgrid models, as described below.

The Bray-Moss-Libby Model for Turbulent Combustion

The BML is a second-order Reynolds-stress model for premixed turbulent combustion (Bray et al. 1981; Libby 1985). Recent overviews are given by Libby and Williams (1994), Bray and Libby (1994), and Bray (1996). BML, in its simplest form, provides a useful framework that is valid for adiabatic, low-Mach-number turbulent flows subject to a global, irreversible combustion reaction. BML uses the progress variable $c(\mathbf{x}, t)$, defined as the

normalized concentration of combustion products. Then the thermodynamic state is

$$\frac{\rho_r}{\rho} = \frac{T}{T_r} = 1 + \tau c, \tag{12–3.7}$$

where $\tau = T_p/T_r - 1$ is considered a constant, and subscripts r and p represent reactants and products, respectively. Pressure fluctuations are assumed small. The progress variable c evolves from $c = 0$, representing completely unburned material, to $c = 1$, representing fully burned material.

Another main assumption of the model is that the flow consists of contributions from the unburned material, the fully burned material, and the part of the material that is undergoing chemical reactions. Further, most of the material is either fully burned or totally unreacted. The BML model treats the regime of very thin flamelets. The mean properties of the flow are calculated in terms of a PDF that has the form

$$\overline{P}(c^o; \mathbf{x}, t) = \alpha(\mathbf{x}, t)\ \delta(c^o) + \beta(\mathbf{x}, t)\ \delta(1 - c^o) + \gamma(\mathbf{x}, t)\ \overline{P}_F(c^o; \mathbf{x}, t), \tag{12–3.8}$$

where c^o is the progress variable, α, β, and γ are the probabilities of there being unburned, fully burned, and partially burned material at $\mathbf{x}$, and $\overline{P}_F(c^o; \mathbf{x}, t)$ is the PDF of this partially reacted mixture. The joint PDF of velocity, u_k, denoted $\overline{P}\left(c^o, u_k^o; \mathbf{x}, t\right)$ also can be partitioned into three parts. The essence of the method, then, is to find a form for and appropriate assumptions leading to the PDF, and then find mean velocities, turbulence intensities, and turbulence stresses in terms of these PDFs. These macroscopic properties averaged over the PDF are used to drive the resolved-scale fluid equations. BML uses Favre-averaged second-order transport equations for high–Reynolds-number turbulent flow with closure found by assuming $\gamma \ll 1$.

Flamelet Models

A fundamental assumption of the flamelet approximation is that chemical reactions are confined to very thin, laminar flamelike regions with thicknesses that are small compared to the scale of turbulence. This is true when chemical time scales are short compared with the small-scale convection and diffusion time scales, that is, when $Da_T \gg 1$. The flames are convected by the flow, and they propagate through unburned regions in the material. These layers may have a well-defined flame structure called a *flamelet*. The finite thickness of a flamelet makes it different from the idealized concept of a *flame sheet*, which is very (infinitely) thin and does not have structure. Excellent reviews of flamelet theories are given by Peters (1986) and Bray and Peters (1994).

In premixed combustion, the flamelets propagate into the unburned mixture normal to the front. Flamelet concepts have been generalized to nonpremixed flames using the transformation to mixture fraction space, as mentioned earlier. In nonpremixed combustion, flamelets are attached to the surfaces. This form of combustion is generally associated with diffusion flames, but may also occur in material that evolves from originally premixed fuel and oxidizer. Here the controlling process is diffusion, which creates a condition under which chemical reactions can occur. The convective and diffusive time scales are generally the same order of magnitude, but the chemical time may be much smaller.

The range for flamelet models is $Da_T \gg 1$ and $Ka_T < 1$, as described in Section 12–4.2. The basic principles for turbulence models for this regime are developed by

modeling the behavior of a flame surface (see the summary in Bray [1996]). In some limits, this flame surface has a finite structure, and in others it is infinitely thin. This surface then propagates through the reactive material, leaving behind changes in reaction products and physical properties. The characteristics of this surface and the changes effected by its passage are predetermined by the properties of the laminar flame. These properties are generally global properties, such as laminar flame speeds as functions of stretching and straining at the flame front.

The mean behavior of this flame surface is usually modeled by a conservation equation for the production and loss of surface area. This equation has terms that must be modeled, as do all of the turbulence models. One current approach is the G equation, where $G(\mathbf{x}, t)$ is a scalar that represents the distance from the flame, and G^o is the instantaneous flame surface (see Section 10–3.3). The flame surface G^o at $G = 0$ then moves through the material and is affected by the fluid dynamics (Williams 1985b; Yakhot 1988; Peters 1992). From a numerical point of view, the G-equation approach could be implemented with one of the front-tracking algorithms described in Chapter 10.

It is interesting to note that the G-equation approach, before it is averaged, is based on a philosophy similar to the induction-parameter models described in Chapter 5. There, a front moved through the material, changing the properties behind the front according to a predetermined prescription of how much energy was released given background conditions. In that case, however, the energy release self-consistently determined the conditions left behind. Here, those conditions could be either determined self-consistently or imposed. Once the G function is treated as a mean quantity and statistics are involved, the basic nature of the model differs considerably.

12–4. Monotone-Integrated Large-Eddy Simulation

Since the early 1970s, the nonlinear monotone algorithms described in Chapters 8 and 9 have been used with increasing confidence to compute a variety of reacting and nonreacting, high–Reynolds-number flows in aerodynamics, combustion, propulsion, atmospheric sciences, and astrophysics. Although originally designed to solve problems with shocks and blast waves, monotone algorithms are now used regularly for subsonic and low-velocity flows with turbulence. The simulations are usually used to resolve the macroscopic scales in a problem, with little concern for the fact that it is not known exactly how the methods behave at the grid-scale cut-off. An implicit assumption has been made, based on many years of experience and successful computations, that "everything is OK."

A background question is asked repeatedly in the fluid-dynamics community: "Why do these methods perform so well computing transitional and turbulent flows when no explicit grid-scale filter or subgrid model is used?" The off-hand answer, for reasons given above, is: "Well, they do reasonably because the large, resolved scales, are the important ones. The small scales are much less important. Numerical diffusion will take care of them." Or you might hear: "Well, there is some nonlinear filter built into the algorithm that seems to be working correctly." Neither answer provides a rigorous explanation of why these methods work or any guidance as to when they might not work so well.

Another fundamental (and perhaps sillier) question, based on misapplied terminology, is still occasionally asked: "Why use monotone algorithms at all when turbulence is so

clearly not monotone? The momentum changes sign locally and frequently in a turbulent flow." The answer to this is straightforward. Monotonicity (or positivity) is a property of the underlying convective derivatives from which the fluid-dynamics equations are built and applies as much to the momentum equations as to the energy and mass density. The convective derivatives representing even turbulent flow should guarantee this fundamental fluid-dynamic property. Turbulent flows have properties that undergo multiple sign changes, that is, show nonmonotone behavior, because of pressure gradients and divergence terms that modify convective transport. Since reasonable implementations of monotone algorithms do not restrict the effects of the divergence and pressure gradient terms, the computed flow correctly develops the small scales and sign changes expected of turbulence. In fact, monotone algorithms do not generate the purely numerical fluctuations and sign changes that may be confused with turbulence, as nonmonotone algorithms may.

In this section, we summarize current evidence supporting the following observation:

> *Flux-limiting monotone algorithms, used for solving (sets of coupled) continuity equations, inherently have a high-frequency filter that maintains physically reasonable behavior near the grid-scale cut-off.*

The consequences of this *MILES* postulate are important: if it is true, we can use monotone algorithms to compute turbulent flows and rely on the inherent properties of the algorithm to provide the correct behavior at the grid-scale cut-off. While MILES does seem reasonable to those who have been simulating high-speed flows, it is not so obvious to the practitioners of RANS and standard LES using nonmonotone algorithms, and perhaps even threatening to those developing new turbulence subgrid-scale models. The key to understanding why this postulate makes sense comes from examining the action of the nonlinear high-frequency filter inherent in the algorithms.

12–4.1. Why MILES Should Work

The nonlinear flux-limiting component of monotone algorithms was described in some detail in Chapter 8. These algorithms (such as FCT, MUSCL, or PPM) were originally developed to solve high-speed compressible flow problems that were usually unsteady, and contained shocks and multiple shock interactions. The basic formulation of monotone algorithms involve various ways in which physical constraints, such as conservation, positivity, and causality, are imposed on the solution. The application of such constraints to the coupled continuity equations that form the Euler and Navier-Stokes equations proceeded independently of the early models of constant-density, homogeneous turbulence and the later extensions to compressible turbulence.

In flux-limiting algorithms for continuity equations, the numerical fluxes preserve the important physical properties of monotonicity, causality, and conservation. Enforcing these properties ensures that sharp gradients, which could be comparable to the grid scale, are maintained and convected correctly. The algorithms operate on the smallest spatial scales in the calculation, as determined by the computational grid. They filter out flow structures smaller than a few grid spacings by spreading and dissipating them on the grid. Given these properties, the idea that monotone algorithms inherently provide a kind of subgrid filter when used to solve the Euler or high–Reynolds-number Navier-Stokes equations is

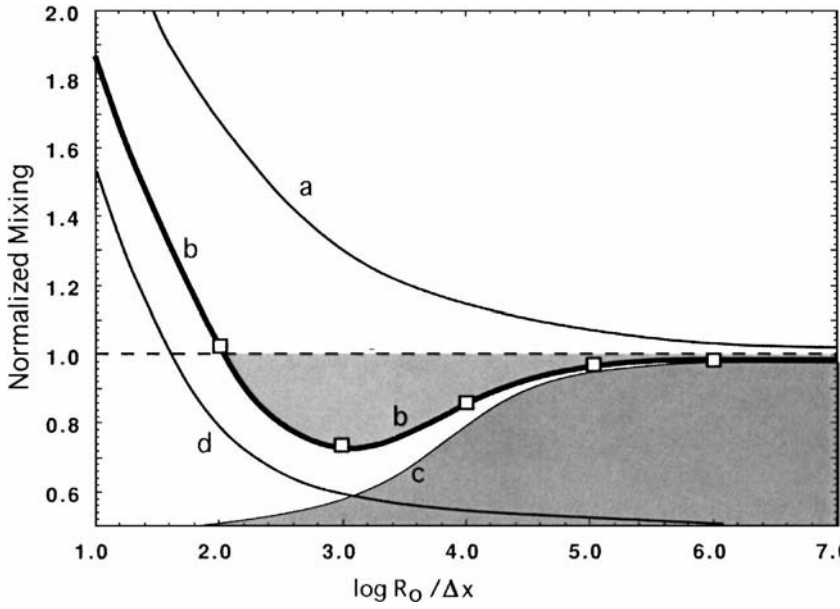

Figure 12.7. Schematic showing how an Euler computation of turbulent entrainment and mixing is expected to converge with increasing resolution. R_o is a scale characteristic of the initial, unmixed configuration, such as the width of a jet; Δx is the computational cell size. (a) Solution obtained with LES using a linearly filtered, diffusive algorithm. (b) Solution obtained with a MILES algorithm, with (c) transport from resolved eddies, and (d) transport from flux limiter.

basically a tautology. The remaining issues that arise with respect to using these methods to describe turbulent flows is: What is the effect of the small-scale cut-off on the large-scale structures? Is the nonlinear filter alone enough to ensure that the physical behavior of the smallest resolved scales does not harm the large scales? Is the filter good enough to correctly dissipate energy at the grid scale?

Figure 12.7 is a schematic diagram showing how a macroscopic quantity, such as entrainment in a shear flow, is expected to behave as a function of computational resolution for conventional LES and MILES in a system with turbulence. Consider, for example, a simulation of a shear layer that becomes unstable and undergoes a transition to turbulence. The vertical axis is a normalized measure of mixing, the volume of entrained fluid at a given time, as computed in a series of simulations at different resolutions. Each point on curves (a) and (b) are taken from a calculation at a different resolution, with the cell spacing decreasing (more highly resolved) to the right. Curve (a), which is completely above the horizontal line, gives an idea of what the entrainment would be computed with the linear-filtered LES algorithms with an added eddy viscosity. Note that an appropriate algorithm to defilter solution (a) could be expected to improve the results. With the nonlinear filtering in a MILES algorithm, the unphysical diffusion does not extend significantly to long wavelengths. This means that additional explicit filtering should not be needed with MILES. Curve (b) is what is expected when a monotone algorithm is used. The ideal, infinite-resolution solution is shown by the horizontal dashed line.

The computations using the monotone algorithm (b) have a minimum in the entrainment at an intermediate resolution. This occurs as a trade-off between two effects, shown

in curves (c) and (d). The minimum occurs when there are still unresolved, active short wavelengths, but the residual numerical diffusion is smaller than the true eddy diffusivity of the turbulent flow. As the resolution is increased, mixing and thus entrainment increase because of resolved eddies. Curve (d) shows increasing mixing and entrainment, a result of very poor resolution, as resolution is decreased. Beyond the minimum in (b), increasing resolution increases entrainment, because reducing the grid-scale numerical filtering has less effect than allowing previously unresolved scales to contribute to the mixing.

The question arises as to why certain other methods, which are not explicitly monotone, give good results. An example of this is the third-order upwind algorithm used by Kawamura and Kuwahara (1984). This may be explained by the fourth-order dissipation term that is used. Apparently, methods have the MILES property when dissipation is strong enough to channel the grid-scale fluctuations smoothly into the unresolved inertial range, and yet the dissipation is sufficiently local or high-order that the resolved scales are not strongly affected.

12–4.2. Tests of the Underlying Concepts

The first discussion of the MILES postulate in the literature was by Boris (1989). The ideas were later expanded and the successful use of monotone algorithms for a several compressible transitional shear flows and turbulent flows was described (Boris et al. 1992a). Here we describe some of these tests that look at the convergence of MILES computations with increasing resolution. One purpose in doing this is to test the underlying ideas shown in Figure 12.7. Another is to examine the subgrid behavior of MILES, and to attempt to quantify the conditions for its validity.

Mixing in Perturbed, Unstable Jets

The concepts and reasoning that led to Figure 12.7 were developed and tested by evaluating the mixing as a function of resolution in a series of simulations of high-speed circular and elliptical jets (Boris et al. 1992a,b). The simulations solved the Euler equations using the monotone LCPFCT algorithm (see Chapter 8). The problem was initialized as a three-dimensional jet with an initially helical perturbation on the interface between the jet and quiescent background materials. As a result of initial perturbation, the jets became unstable and vortices formed at the interface between the jet and the background material. These computations were performed using an FCT (monotone) algorithm on a series of different grids ranging in size from 32^3 to 256^3 computational cells. Figure 12.4 shows cross-sections through the center of the jet in the early mixing phase of a simulation with 256^3 computational cells. The small-scale secondary instabilities associated with the beginning of the turbulent cascade increased the amount of entrained jet material even as the large-scale, primary Kelvin-Helmholtz instability continued growing.

One set of convergence tests on the elliptical-jet configuration is summarized in Figure 12.8. The figure shows that, as the resolution is increased, the amount of entrainment of material approaches a limiting value. These results are presented in a way similar to that shown in Figure 12.7. As the resolution ($R_o/\Delta x$) increases, the entrainment converges quickly, despite the existence of a wide range of unresolved small scales in the simulated problem and the lack of an explicit, added subgrid turbulence model. This may be

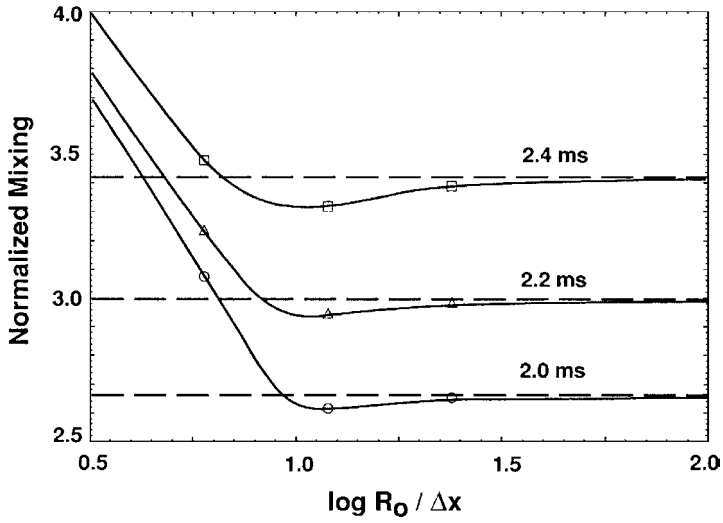

Figure 12.8. Mixing as a function of resolution from MILES simulations of an elliptical jet (Boris et al. 1992a). As the computational cell size is decreased, the amount of material entrained does in fact approach a limiting value, as shown in Figure 12.7. This is shown for three different times in the simulation.

considered evidence that the built-in high-frequency filter in FCT is behaving in a physically correct way.

Evolution of Compressible Turbulence

Similarly motivated tests were performed using a different monotone algorithm, PPM (see Chapters 8 and 9). Porter, Pouguet, and Woodward (1992a,b, 1994) studied a compressible turbulent flow in which shear layers, formed through shock interactions, are distributed inhomogeneously throughout the flow. They solved the Euler equations and, where possible, compared the results to equivalent Navier-Stokes solutions. A series of successively more resolved simulations, using grids of 64^3, 128^3, 256^3, and 512^3 cells, were performed in a computation domain representing a periodic, cubical box, and the flow properties were Fourier analyzed to find the turbulence spectrum. These studies of turbulence in a periodic, cubic domain are similar in some ways to those direct numerical simulations described in Section 12–1, except here the flow is compressible, and the tests are performed to determine the convergence properties of the algorithm as the resolution is increased.

The results that are important to our discussion here are typified by those shown in Figure 12.9. Even without an explicit subgrid turbulence model, the Euler equation solutions are well characterized by the $5/3$ power envelope in an inertial range and fill out the spectrum to progressively smaller scales as the grid resolution is increased. Porter and colleagues showed that the computations converge as resolution increases and approach the solution obtained by very high-resolution Navier-Stokes computations of the identical physical problem. In this case, the Kolmogorov $k^{-5/3}$ spectrum was expected to exist only as a transient because the flow is decaying rather than being driven at long wavelengths. Instead, it appeared as an envelope to spectra from a series of MILES calculations performed with increasing resolution. This behavior has been seen for several

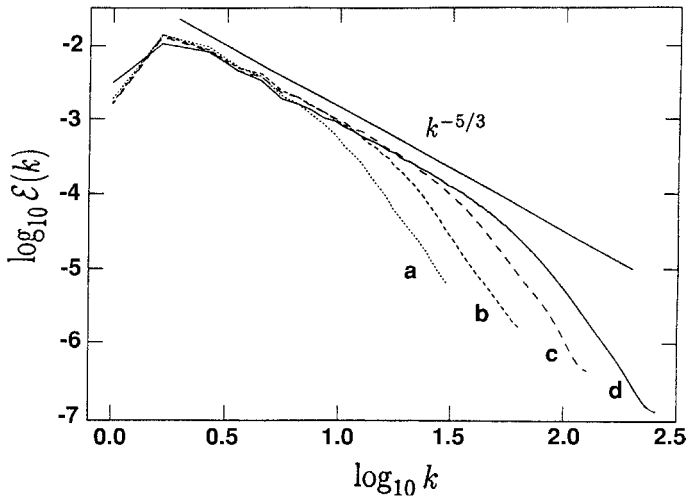

Figure 12.9. Convergence of monotone-integrated LES to the $k^{-5/3}$ Kolmogorov spectrum illustrated using data from four three-dimensional MILES simulations using the PPM algorithm. Increasing resolution extends the length of the inertial range with little effect on the large, energy-containing scales (Porter et al. 1994). (a) 64^3, (b) 128^3, (c) 256^3, and (d) 512^3.

two-dimensional configurations and the same behavior is also seen in three dimensions (Porter et al. 1994). The converging spectra in this work are support of the validity and usefulness of the effective subgrid model provided by the flux limiter in PPM.

12–4.3. Empirical Evidence that MILES Works for Complex Flows

Part of the evidence that MILES is a good approach to LES lies in the body of successful simulation results produced for more complex problems, for which the geometry or physics complicates or even precludes other LES or turbulence modeling approaches. MILES-based computations have now produced results that are in generally good agreement with experiments, observations, other simulations, and known analytic solutions. Below we describe several examples of calculations that have added credibility to the MILES postulate.

Transitional Flows: Jet Flows and Splitter Plates

A series of two- and three-dimensional simulations of subsonic shear flows by Grinstein, Fureby, and coworkers examined the evolution of large-scale coherent structures in jets undergoing transition to turbulence (see, e.g., Grinstein et al. [1996] and Fureby and Grinstein [1999], and references therein). These calculations used FCT-based codes with no explicit subgrid models. Even though the high-Re experiments on these flows are turbulent throughout much of the regime, the simulations reproduced the important features of the flow despite the unresolved small-scale structure. For example, comparisons of experimental and calculated data agreed qualitatively and quantitatively for both instantaneous and time-averaged quantities such as entrainment, jet-spreading rates, and vortex pairing.

Recent three-dimensional tests (Fureby and Grinstein 1999) compared solutions of turbulent flows with and without an additional subgrid model. This work showed that the large-scale features of a transitional free jet were independent of the particular subgrid

model used, when *Re* is high enough and a range of large scales are resolved to carry the computation into the inertial range.

Rayleigh-Taylor Instability and Inertial Confinement Fusion

When a high-powered laser beam interacts with a stationary thin foil made of polystyrene, the surface of the foil forcibly evaporates (ablates), accelerating the foil away from the laser. This acceleration subjects the polystyrene and other materials layered behind this foil to a strong Rayleigh-Taylor instability in which the pressure gradient and the density gradient are opposed. These instabilities are excited by imperfections in the flat foil and asymmetries in the laser beam. The result of this instability is that the accelerated material surface becomes very convoluted and eventually breaks. This process occurs when a laser beam is used to implode pellets of deuterium and tritium (D-T), which is part of an effort to produce laboratory-scale nuclear fusion (Gardner and Bodner 1986). This is an extremely complex physical problem characterized by *Re* in the range 10^4 to 10^6. Thus a spectrum of compressible turbulence is excited at the density discontinuity between the plastic and the D-T fuel. This complicated mixing interaction is another class of compressible, reactive-flow problems for which FCT-based MILES codes have been used.

An extensive series of multidimensional calculations with realistic equations of state for the materials, electron thermal conduction, and radiation transport (but without added subgrid turbulence models), were validated extensively with experiments. These models are now used on a production basis for designing experiments (Emery, Gardner, and Boris 1982; Emery, Dahlburg, and Gardner 1988; Dahlburg and Gardner 1990; Dahlburg et al. 1995). The fact that the comparisons between experimental and computed data show good agreement supports the idea that the underlying fluid-dynamics algorithm properly accounts for the unresolved, small-scale motions of the highly nonuniform compressible medium.

Low–Mach-Number Bluff-Body Flows

A group of researchers led by Kuwahara (see, e.g., Tsuboi, Tamura, and Kuwahara [1989]; Suzuki and Kuwahara [1989]; Tamura and Kuwahara [1989]) have performed simulations of turbulent flows using a third-order upwind method. This method seems to be nearly monotone in its behavior, which is explained by its having a fourth-order dissipation term. The method, without any extra LES filtering or eddy-transport terms, has been used to compute vortex shedding and vortex separation, in both two and three dimensions for very high Reynolds-number flows. Significant vortex shedding and separation occur in these flows at scales that are even smaller than those resolved by the fine grids that are used. Small-scale vortices, generated in a well-resolved boundary layer, are convected into regions of the grid where they cannot be resolved. The numerical model automatically filters these unresolvable fluctuations, apparently without degrading the solution or having any damaging effects on the computed large-scale quantities. Reynolds numbers as high as 10^6 have been simulated this way. One example of the success of this approach is in computing the drag crisis, which is the change in aerodynamic drag on a circular cylinder when the Reynolds number becomes so large that the boundary layer on the cylinder changes from laminar to turbulent. This phenomenon was reproduced in a computation that did not include any subgrid model. A similar series of computations was performed

for a wide splitter plate using monotone FCT algorithms (Boris et al. 1989; Grinstein, Boris, and Griffin 1991).

Another series of computations by Fureby and coworkers (Fureby and Möller 1995; Fureby 1996; and Möller, Lundgren, and Fureby 1996) simulated the geometry of a channel with a bluff triangular-shaped body. This geometry was based on an experimental chamber that was used for studies of air flows around the body and for reacting propane-air flows. The unreacted gas flowed at 20 m/s into a rectangular chamber containing the triangular obstacle designed to stabilize and hold the flame. In addition to the experiments, an extensive series of calculations was performed to test and compare a number of the different LES subgrid models with MILES. To ensure monotonicity, a flux-limiter was used with the convection algorithm. In the case of reacting flows, additional terms were added to the equations to simulate the chemical reactions with subsequent heat release and radiation transport. Again, it was found that the simulations produced substantially the same results for the large-scale features of the flow, whether or not the subgrid model was included.

Decay of Isotropic Turbulence

Knight and coworkers developed a finite-volume unstructured-grid method that has high-order dissipation and flux-limiting features (Knight et al. 1998; Okong'o and Knight 1998). They tested the MILES performance of their unstructured-grid model by comparing simulations based on the Euler equations to benchmark experiments (Comte-Bellot and Corrsin 1971) that measured the decay of the turbulent kinetic energy and the time evolution of the spectra in the decay of isotropic turbulence. Their conclusions were that the code, without an additional subgrid-scale model, gave a good representation of these quantities.

Study of the same problem by Fureby et al. (1997) compared DNS to LES of homogeneous forced and decaying turbulence in a box. The extensive series of tests included MILES and a variety of subgrid-scale models. The basic conclusion was that the large-scale features of the flow were independent of the subgrid model as long as the spatial resolution was adequate. Later work by Fureby and Grinstein (1999) compared several different types of MILES algorithms by varying the order of the method and the particular flux limiter. Their conclusions are similar but provide more guidance and some evaluation of using MILES in Navier-Stokes and Euler simulations.

12–4.4. Conclusions and Speculations

We conclude this section by considering how monotone algorithms function as a subgrid model. The metrics are the properties of an ideal subgrid model, listed in Section 12–2. An LES subgrid model should:

Be Sufficiently General. Existing subgrid models, until recently, have been limited to constant density, incompressible, nonreacting flows on uniformly spaced meshes. Current developments of LES for compressible and reactive flows, even with Favre averaging and filtering, have a number of nonlinear closure terms and require correspondingly more models with parameters to be calibrated. These difficulties, at least to first order, do not apply to MILES approaches.

Be Conservative. Conservation is usually built into monotone algorithms. In fact, conservation is one feature that helps ensure monotonicity.

Minimally Contaminate the Resolved Scales. Inaccuracy at the grid scale and the effects of filters are concerns for standard LES models, but not major concerns for MILES. Since the dissipation is minimal and flux limiting is highly localized in monotone algorithms, the well-resolved long wavelengths are minimally contaminated. Analyses of FCT has shown that there is an effective dissipation that scales roughly as the fourth power of the spatial scale (Book, Patnaik, and Grinstein 1991; Grinstein and Guirguis 1992). This means that a flow structure ten times larger than a structure at the grid scale is subject to roughly two orders of magnitude less residual dissipation than it would experience from a method that uses conventional LES with a subgrid turbulence model. Thus the large structures should be convected accurately for Reynolds numbers up to 100 times larger than the dissipation applied to the small scales would indicate.

Model Onset and Other Transient Laminar Phenomena. MILES creates subgrid dissipation only when the algorithm detects unresolvable flow structures near the grid scale. Therefore, MILES does not affect laminar, well-resolved flow. MILES can also provide a direct, local measure of the subgrid-scale structure by considering the computed but unused antidiffusive fluxes.

Couple Large-Scale and Subgrid Motions. MILES algorithms automatically feed back the mean subgrid transport into the resolved scales, both through conservation terms (unresolved kinetic energy becomes heat) and the effects of the flux limiter at the grid scale. The mean subgrid transport appears in monotone methods as a residual local dissipation left by the flux-limiting process. MILES provides for deterministic backscatter from the short wavelengths to the resolved scales because of the intermittent, nonlinear, and localized nature of the flux-correction algorithm. In either LES or MILES, however, the feedback to the large scales is missing phase information about the unresolved small scales. Because unresolved subgrid fluctuations can act as infinitesimal triggers of instability, a small random source term in the equations would seem to be a useful addition when there is subgrid structure.

Ensure Smooth Connectivity. Conventional LES has severe problems with spatially and temporally varying grids. Even when the grid is uniform, the formulation of turbulence subgrid models does not explicitly account for the algorithm-specific grid-scale truncation errors. MILES does not have these problems. This is probably the most attractive and compelling aspect of using MILES. Even though the action of the intrinsic nonlinear filter in FCT (and probably other monotone algorithms) is problem and grid dependent, the solution of the underlying partial differential equations rigorously converges as the resolution is increased (Ikeda and Nakagawa 1979).

Treat Complicated Reactions and Flows. The framework adopted for monotone algorithms treats the physical flow variables in a way that allows other terms, representing other physical processes, to be added (see Chapter 11). Further, because of the conservative, local nature of the algorithms, new source terms and boundary conditions may also be added.

Be Economical. Monotone CFD algorithms for solving the conservation equations are relatively easy to program and are flexible enough to be used when the geometry is complex. Therefore, they are very economical. Their economy is further enhanced by their very high resolution. They require almost an order of magnitude fewer cells in three dimensions than linear convection algorithms to obtain comparable accuracy.

Since MILES methods have not usually been thought of as LES models, there are areas where further interpretation and verification are needed. Inverting this built-in filter *(defiltering)* is always a problem with LES models. This is not practical in MILES because the filter is nonlinear. This inversion should not be necessary in MILES for quantities that depend on the well-resolved scales of motion. Conversely, the monotone filters can be applied to experimental data or to theoretical models for exact comparison.

As introduced in Section 12–3.3, Fureby and Möller (1995) and Möller et al. (1996) studied a premixed propane-air flame stabilized on a bluff body. They compared experimental data to simulations using a number of different subgrid-scale models, including MILES. Their conclusion was that the solution was "resilient" to the subgrid model: MILES works, as long as the cut-off wavenumber determined by the grid scale is in the inertial range. MILES reproduced the global, experimentally measured features (densities, temperatures, etc.) at least as well as any of the other LES models used. Thus there is some hope for simplicity.

A recent series of studies of MILES of both transitional jet flows and isotropic homogeneous turbulence in a box compared the effects of various subgrid models in conventional LES and various flux limiters in montone methods (Fureby and Grinstein 1998, 1999). These simulations provided even stronger confirmation that MILES provides a valid subgrid model as long as enough of the inertial range is resolved. Their analysis of the stress tensors implicit in MILES shows that MILES should work for turbulent channel flow, depending on the choice of flux limiter in the monotone algorithm. Further, this analysis puts MILES on a firm theoretical footing by relating the expected subgrid stress-tensor terms to features of the specific flux limiter used in the monotone algorithm.

This section has made some efforts to describe what MILES can do. It is appropriate to be as clear as possible about what it cannot do and what might have to be added. We know that MILES will not work well unless enough of the large scales are resolved to capture the bulk of the turbulent kinetic energy of the flow. MILES may not predict the right result if the calculation is two dimensional, but the turbulence is dominant and three dimensional. Some kind of phenomenology should be included to describe the third dimension. MILES, as currently formulated, does not include stochastic backscatter, which would have to be represented by an additional model. Some added eddy viscosity probably should be added above the amount provided by the algorithms to account for the missing fluctuating-phase components of the subgrid fields. This is true for both MILES and other LES. MILES is not a fluid-structure interactions model, therefore additional physics should be included to describe boundary layers. Adding viscosity will do this if the resolution at the body surface is fine enough, but the computations will become very expensive. Indeed, it is unreasonable for MILES, or any other subgrid model, to describe physical processes that are not even in the equation set.

Finally, even though some reactive MILES computations give reasonable answers, we cannot expect that this should always be the case. MILES should take care of the proper flow of energy through the inertial range, and the decay at the grid-scale cut-off. But it is not a chemistry model, and so can only handle the fluid parts of the problem. The extent to which other types of models need to be included for exothermic flows with complex chemical-reaction mechanisms is a topic of current research.

Turbulence is a difficult phenomenon to model accurately and reactive turbulence is very difficult. When a topic is this complex, it is always worthwhile to consider new ideas and potentially useful concepts. It is equally important to remember that:

Perfection is the enemy of good enough.

Increasing accuracy in an algorithm is usually bought at the price of economy and generality. Therefore, it is usually best to use simple models that have the correct physical behavior in the limits, and that have physically calibrated models governing the behavior between the limits.

REFERENCES

Baritaud, T., T. Poinsot, and M. Baum. 1996. *Direct numerical simulation for turbulent reacting flows*. Paris: Éditions Technip.

Blaisdell, G.A., N.N. Mansour, and W.C. Reynolds. 1993. Compressibility effects on the growth and structure of homogeneous turbulent shear flow. *Journal of Fluid Mechanics* 256:443–485.

Book, D.L., C. Li, G. Patnaik, and F.F. Grinstein. 1991. Quantifying residual numerical diffusion in flux-corrected transport algorithms. (*Journal of Scientific Computation* 6:323–343).

Borghi, R. 1988. Turbulent combustion modeling. *Progress in Energy and Combustion Science* 14:245–292.

Boris, J.P. 1989. On large eddy simulations using subgrid turbulence models. In *Wither turbulence? Turbulence at the crossroads*, ed. J.L. Lumley, *Lecture Notes in Physics* 257:344–353. Berlin: Springer-Verlag.

Boris, J.P., O.M. Griffin, J.D. Baum, J.H. Gardner, F.F. Grinstein, K. Kailasanath, E.S. Oran, and T.R. Young. 1989. Vortex shedding and base pressure behind a wide splitter plate. In *Pressure vessels and piping–flow induced vibration* 154, eds. M.K. Au-Yang, S.S. Chen, S. Kaneko, and R. Chilukuri, ASME Book No. HOO469, 147–154, NY, NY: American Society of Mechanical Engineers.

Boris, J.P., F.F. Grinstein, E.S. Oran, and R.L. Kolbe. 1992a. New insights into large eddy simulation. *Fluid Dynamics Research* 10:199–228.

Boris, J.P., R. Hubbard, E. Oran, S. Slinker. 1992b. Enhanced mixing from shock-generated turbulence in dusty air. In *Proceedings of the 18th International Symposium on Shock Waves*, ed. K. Takayama, 553–558. Sendai, Japan: Institute of Fluid Science, Tohoku University.

Bray, K.N.C. 1996. The challenge of turbulent combustion. In *Proceedings of the Twenty-Fifth Symposium (International) on Combustion*, 1–26. Pittsburgh, Pa: The Combustion Institute.

Bray, K.N.C., and P.A. Libby. 1994. Recent developments in the BLM model of premixed turbulent combustion. In *Turbulent reacting flows*, eds. P.A. Libby and F.A. Williams, 115–152. San Diego: Academic Press.

Bray, K.N.C., P.A. Libby, G. Masuya, and J.B. Moss. 1981. Turbulence production in premixed turbulent flames. *Combustion Science and Technology* 25:127–140.

Bray, K.N.C., and Peters N. 1994. Laminar flamelets in turbulent flames. In *Turbulent reacting flows*, eds. P.A. Libby, and F.A. Williams, 63–114. San Diego: Academic Press.

Comte-Bellot, G., and S. Corrsin. 1971. Simple Eulerian time correlation of full- and narrow-band velocity signals in grid-generated, isotropic turbulence. *J. Fluid Mech.* 48:273–337.

Cooley, J.W., and J.W. Tukey. 1965. An algorithm for the machine computation of complex fourier series. *Mathematics of Computation* 19:297–301.

Dahlburg, J.P., and J.H. Gardner. 1990. Ablative Rayleigh-Taylor instability in three dimensions. *Physical Review A* 41:5695–5698.

Dahlburg, J.P., M. Klapisch, J.H. Gardner, C.R. DeVore, A.J. Schmitt, and A. Bar-Shalom. 1995. Radiating plasma structures in ablating, laser-produced plasmas. *Journal of Quantitative Spectroscopy and Radiation Transfer* 54:113–121.

Domaradzki, J.A. 1992. Analysis of subgrid-scale eddy viscosity with the use of the results from direct numerical simulations. *Physics of Fluids A* 4(9):2037–2045.

Domaradzki. J.A., R.W. Metcalfe, R.S. Rogallo, and J.J. Riley. 1987. Analysis of subgrid-scale eddy viscosity with the use of the results from direct numerical simulations. *Physical Review Letters* 58:547–550.

Dopazo. C. 1994. Recent developments in PDF methods. In *Turbulent reacting flows*, eds. P.A. Libby and F.A. Williams, 375–474. San Diego: Academic Press.

Emery, M.H., J.P. Dahlburg, and J.H. Gardner. 1988. Linear and nonlinear evolution of the Rayleigh-Taylor Instability in laser ablatively accelerated targets. *Laser Interaction and Related Phenomena* 8:401–416. New York: Plenum Publishing.

Emery, M.H., J.H. Gardner, and J.P. Boris. 1982. Rayleigh-Taylor and Kelvin-Helmholtz instabilities in targets accelerated by laser ablation. *Physical Review Letters* 48:677–680.

Erlebacher, G., M.Y. Hussaini, H.O. Kreiss, and S. Sarkar. 1990. The analysis and simulation of turbulence. *Theor. Comput. Fluid Dyn.* 2:73–95.

Feiereisen, W.J., W.C. Reynolds, J.H. Ferziger. 1981. *Numerical simulation of a compressible homogeneous turbulent shear flow*. Rep. TF-13. Stanford, Calif: Thermosciences Division, Department of Mechanical Engineering, Stanford University.

Fox, D.G., and D.K. Lilly. 1972. Numerical simulation of turbulent flows. *Rev. Geophys. Space Phys.* 10:51–72.

Fureby, C. 1996. On subgrid scale modeling in large eddy simulations of compressible fluid flow. *Physics of Fluids* 8:1301–1311.

Fureby, C., and F.F. Grinstein. 1998. High-Reynolds number large-eddy simulation of free shear flows. In *Sixteenth International Conference on Numerical Methods in Fluid Dynamics*, ed. C.-H. Bruneau, 165–170. New York: Springer.

———. 1999. Monotonically integrated large eddy simulation of free shear flows. *AIAA Journal* 37:544–556.

Fureby. C., and S.I. Möller. 1995. Large eddy simulation of reacting flows applied to bluff body stabilized flames. *AIAA Journal* 33:2339–2347.

Fureby. C., G. Tabor, H.G. Weller, and A.D. Gosman. 1997. A comparative study of subgrid scale models in homogeneous isotropic turbulence. *Physics of Fluids* 9:1416–1429.

Furukawa, J., K. Okamoto, and T. Hirano. 1996. *Turbulence characteristics within the local reaction zone of a high-intensity turbulent premixed flame.* In *Twenty-Sixth Symposium (International) on Combustion*, 405–412. Pittsburgh, Pa.: The Combustion Institute.

Gardner, J.H., and S.E. Bodner. 1986. High-efficiency targets for high-gain inertial confinement fusion. *Physics of Fluids* 29:2672–2678.

Grinstein. F.F., J.P. Boris, and O.M. Griffin. 1991. Passive pressure-drag control in a plane wake. *AIAA Journal* 29:1436–1442.

Grinstein. F.F., and R.H. Guirguis. 1992. Effective viscosity in the simulation of spatially evolving shear flows with monotonic FCT models. *Journal of Computational Physics* 101(1):165–175.

Grinstein, F.F., E.J. Gutmark, T. Parr, D. Hanson-Parr, and U. Obeysekare. 1996. Streamwise and spanwise vortex interaction in an axisymmetric jet: A computational and experimental study. *Physics of Fluids* 8:1515–1524.

Herring, J.R. 1973. Statistical turbulence theory and turbulence phenomenology. In *Proceedings of the Langley Working Conference on Free Turbulent Shear Flows*, NASA SP 321, 41–66, Hampton Va: Langley Research Center.

———. 1985. An introduction and overview of various theoretical approaches to turbulence. In *Theoretical approaches to turbulence*, eds. D.L. Dwoyer, M.Y. Hussaini, and R.G. Voigt, 73–90. New York: Springer.

Hinze, J.O. 1975. *Turbulence*. 2nd ed. New York: McGraw-Hill.

Ikeda, T., and T. Nakagawa. 1979. On the SHASTA FCT algorithm for the equation $\partial\rho/\partial t + \partial/\partial x(v(\rho)\rho) = 0$. *Mathematics of Computation* 33:1157–1169.

Jones, W.P. 1994. Turbulence modelling and numerical solution methods for variable density and

combusting flows. In *Turbulent reacting flows*, eds. P.A. Libby, and F.A. Williams, 309–374. San Diego: Academic Press.

Kawamura, T., and K. Kuwahara. 1984. Computation of high Reynolds number flows around a circular cylinder with surface roughness. AIAA Paper 84-0340. Reston, Va.: American Institute of Aeronautics and Astronautics.

Kleiser, L., and T.A. Zang. 1991. Numerical simulation of transition in wall-bounded shear flows. *Annual Reviews of Fluid Mechanics* 23:495–537.

Knight, D., G. Zhou, N. Okong'o, and V. Shukla. 1998. Compressible large eddy simulation using unstructured grids. AIAA Paper 98-0535. Reston, Va.: American Institute of Aeronautics and Astronautics.

Kollman, W. 1990. The PDF approach to turbulent flow. *Theoretical and Computational Fluid Dynamics* 1:249.

Lam, S.H. 1992. On RNG theory of turbulence. *Physics of Fluids A* 4:1007–1017.

Landau. L.D., and E.M. Lifshitz. 1959. *Fluid mechanics*. New York: Pergamon Press.

Leith. C.E. 1990. Stochastic backscatter in a subgrid-scale model: Plane shear mixing layer. *Physics of Fluids A* 2:297–299.

Leith, C.E., and R.A. Kraichnan. 1972. Predictability of turbulent flow. *Journal of Atmospheric Science* 29:1041–1058.

Lesieur, M. and O. Métais. 1996. New trends in large-eddy simulations of turbulence. *Annual Reviews of Fluid Mechanics* 28:45–82.

Libby, P.A. 1985. Theory of normal premixed flames revisited. *Progress in Energy and Combustion Science* 11:83–96.

Libby, P.A., and F.A. Williams (eds.). 1980. *Turbulent reacting flows*. Topics in Applied Physics No. 44. New York: Springer-Verlag.

Libby, P.A., and F.A. Williams. 1994. Fundamental aspects and a review. In *Turbulent reacting flows*, eds. P.A. Libby and F.A. Williams, 1–62. San Diego: Academic Press.

Liñán, A. and F.A. Williams. 1993. Turbulent combustion. In *Fundamental aspects of combustion*, 111–152. New York: Oxford.

Lorentz, E.N. 1969. The predictability of a flow which possesses many scales of motion. *Tellus* 21:289–307.

Lumley, J.L. (ed.). 1990. *Whither turbulence? Turbulence at the crossroads*. Berlin: Springer-Verlag.

McComb, W.D. 1985. Renormalization group methods applied to the numerical simulation of fluid turbulence. In *Theoretical approaches to turbulence*, eds. D.L. Dwoyer, M.Y. Hussaini, and R.G. Voigt, 187–208. New York: Springer.

McMurtry, P.A., and P. Givi. 1991. Spectral simulations of reacting turbulent flows. In *Numerical Approaches to Combustion Modeling*, eds. E.S. Oran and J.P. Boris, *Progress in Astronautics and Aeronautics* 135, 257–303. Reston, Va.: American Institute of Aeronautics and Astronautics.

McMurtry, P.A., W.-H. Jou, J.J. Riley, and R.W. Metcalfe. 1986. Direct numerical simulations of a reacting mixing layer with chemical heat release. *AIAA Journal* 24:962–970.

Moin, P., and K. Mahesh. 1998. Direct numerical simulation: A tool in turbulence research. *Annual Reviews of Fluid Mechanics* 30:539–578.

Möller, S.I., E. Lundgren, and C. Fureby. 1996. Large eddy simulation of unsteady combustion. In *Twenty-Sixth Symposium (International) on Combustion*, 241–248. Pittsburgh, Pa.: The Combustion Institute.

O'Brien, E.E. 1980. The probability density function (pdf) approach to reacting turbulent flows. In *Turbulent reacting flows*, eds. P.A. Libby and F.A. Williams, 185–218. New York: Springer-Verlag.

Okong'o, N., and D. Knight. 1998. Accurate three-dimensional unsteady flow simulation using unstructured grids. AIAA Paper 98-0787. Reston, Va.: American Institute of Aeronautics and Astronautics.

Orszag, S.A. and G.S. Patterson. 1972. Numerical simulation of three-dimensional homogeneous isotropic turbulence. *Physical Review Letters* 28:76–79.

Peters, N. 1986. Laminar flamelet concepts in turbulent combustion. In *Twenty-First Symposium (International) on Combustion*, 1231–1260. Pittsburgh, Pa.: The Combustion Institute.

———. 1992. A spectral closure for premixed turbulent combustion in the flamelet region. *Journal of Fluid Mechanics* 242:611–629.

Picone, J.M., and J.P. Boris. 1983. Vorticity generation by shock propagation through bubbles in a gas. *Journal of Fluid Mechanics* 189:23–51.

Piomelli, U., W.H. Cabot, P. Moin, and S. Lee. 1991. Subgrid-sale backscatter in turbulent transitional flows. *Physics of Fluids A* 6:1766–1771.

Poinsot, T., S. Candel, S., and A. Trouvé. 1996. Applications of direct numerical simulation to premixed turbulent combustion. *Progress in Energy and Combustion Science* 21:531–576.

Pope, S.B. 1985. PDF methods for turbulent reactive flows. *Progress in Energy and Combustion Science* 11:119–192.

———. 1990. Computations of turbulent combustion: Progress and challenges. In *Proceedings of the Twenty-Third Symposium (International) on Combustion*, 591–612. Pittsburgh, Pa.: The Combustion Institute.

———. 1994. Lagrangian PDF methods for turbulent flows. *Annual Reviews of Fluid Mechanics* 26:23–63.

Porter, D.H., A. Pouquet, and P.R. Woodward. 1992a. Three-dimensional supersonic homogeneous turbulence: A numerical study. *Physical Review Letters* 68:3156–3159.

———. 1992b. A numerical study of supersonic turbulence. *Theoretical and Computational Fluid Dynamics* 4:13–49.

———. 1994. Kolmogorov-like spectra in decaying three-dimensional supersonic flows. *Physics of Fluids* 6:2133–2142.

Rai, M.M., T.B. Gatski, and G. Erlebacher. 1995. Direct simulation of spatially evolving compressible turbulent boundary layers, AIAA Paper 95-0583. Reston, Va.: AIAA.

Reynolds, W.C. 1976. Computation of turbulent flows. *Annual Reviews of Fluid Mechanics* 8:183–208.

Riley, J.J., and R.W. Metcalfe. 1980. Direct numerical simulation of a perturbed, turbulent mixing layer, AIAA Paper No. 80-0274. Reston, Va.: AIAA.

Riley, J.J., R.W. Metcalfe, and S.A. Orszag. 1986. Direct numerical simulation of chemically reacting mixing layers. *Physics of Fluids* 29:406–422.

Rogallo, R.S. 1981. Numerical experiments in homogeneous turbulence, NASA TM-81315, Langley, Va.: NASA Langley Research Center.

Rogallo, R.S., and P. Moin. 1984. Numerical simulation of turbulent flows. *Annual Reviews of Fluid Mechanics* 16:99–137.

Smagorinsky, J. 1963. General circulation experiments with the primitive equations. I. The basic experiment. *Monthly Weather Review* 91:99–164.

Smith, L.M., and S.L. Woodruff. 1998. Renormaliation-group analysis of turbulence. *Annual Reviews of Fluid Mechanics* 30:275–310.

Speziale, C.G. 1991. Analytical methods for the development of Reynolds-stress closures in turbulence. *Annual Review of Fluid Mechanics* 23:107–157.

Suzuki, M., and K. Kuwahara. 1989. Stratified flow past a bell-shaped hill, AIAA 89–1824. Reston, Va.: American Institute of Aeronautics and Astronautics.

Tamura, T., and K. Kuwahara. 1989. Numerical analysis on aerodynamic characteristics of an inclined square cylinder, AIAA 89–1805. Reston, Va.: American Institute of Aeronautics and Astronautics.

Tennekes, H. and J.L. Lumley. 1972. *A first course in turbulence*, Cambridge, Mass.: MIT Press.

Tsuboi, K., T. Tamura, K. Kuwahara. 1989. Numerical study of vortex induced vibration of a circular cylinder in high reynolds number flow, AIAA 89–0294. Reston, Va.: American Institute of Aeronautics and Astronautics.

Vervisch, L., and T. Poinsot. 1998. Direct numerical simulation of non-premixed turbulent flames. *Annual Reviews of Fluid Mechanics* 30:655–691.

Vreman, A.W., N.D. Sandham, and K.H. Luo. 1996. Compressible mixing layer growth rate and turbulence characteristics. *Journal of Fluid Mechanics* 320:235–258.

Vuillermoz, P., and E.S. Oran. 1995. Mixing regimes in a spatially confined two-dimensional compressible mixing layer. *Proceedings of the Royal Society, London, A*, 449:351–380.

Williams, F.A. 1985a. *Combustion theory*. 2nd ed. Redwood City, Calif.: Addison-Wesley.

———. 1985b. Turbulent combustion. In *The Mathematics of Combustion*, ed. J.D. Buckmaster, 97–125. Philadelphia: SIAM.

Yakhot, V. 1988. Propagation velocity of premixed turbulent flames. *Combustion Science and Technology* 60:191–214.

Yakhot, V., and S.A. Orszag. 1986a. Renormalization-group analysis of turbulence, I. Basic theory. *Journal of Scientific Computing* 1:3–51.

———. 1986b. Renormalization-group analysis of turbulence. *Physical Review Letters* 57:1722–1724.

Yeung, P.K., and J.G. Brasseur, 1991. The response of isotropic turbulence to isotropic and anisotropic forcing at the large scales. *Physics of Fluids A* 3(5):884–896.

Zeldovich, Ya.B., G.I. Barenblatt, V.B. Librovich, and G.M. Makhviladze. 1985. *The mathematical theory of combustion and explosions*. New York: Consultants Bureau.

13

Radiation Transport and Reactive Flows

The electronic, atomic, and molecular motions associated with internal energy cause materials to emit and absorb electromagnetic radiation continuously. Electromagnetic radiation spans a wide spectrum, ranging from radio waves to cosmic rays, and it is an important energy-transport mechanism. As such, it is an important physical effect and material diagnostic in reactive-flow systems, such as black-hole accretion disks, stellar interiors, large-scale fires, small-scale laboratory flames, rocket propulsion, hypersonic shock layers, and laser-matter interactions. For example, in forest fires or furnaces, radiation can cause ignition at widely separated regions by a phenomenon called *flashover*. Flashover occurs when the radiation from one combustion region heats a distant surface until it ignites. Radiation can also be important in engine combustion chambers, where temperatures reach two or three thousand degrees Kelvin. Soot particles formed by combustion processes emit and absorb radiation, thereby changing the heat balance and thus the buoyancy of the products.

The energy-exchange mechanisms for conduction and convection differ fundamentally from those of radiation. For example, emitted radiation depends very sensitively on the material temperature and becomes more important as the temperature increases. The net radiant energy transferred generally depends on differences of the absolute temperatures raised to the fourth power, following the Stefan-Boltzmann law. The energy-exchange mechanisms for convection and conduction usually depend linearly on the temperature difference. Another important difference between radiation transport and transport by conduction or convection is that radiant energy, carried by photons, can be transported in a vacuum as well as in a material medium. In convection and conduction, energy is transported by the material medium.

Radiation transport is a major scientific field in its own right. There have been many books and treatises written on it. Many methods have been developed both to approximate the basic equations of radiation transport or to solve them directly, and there has been a great deal of material written using the results for analysis and prediction. Thus there is no way that a single chapter can do justice to describing the variety of methods that have been developed for radiation transport and their applications in reactive-flow simulations. Our purpose in this chapter is to provide a starting point for those interested in reactive-flow problems for which neglecting radiative transport cannot be justified. This chapter briefly

describes the physical basis of radiation transport and uses this to introduce the equations of radiative transfer. From these equations, we can derive an expression for the radiant-energy flux, $\mathbf{q}_r$, introduced in equation (2–1.4) and discussed briefly in Section 2–2.4. We then discuss several limiting cases. These limiting cases, however, are not general enough for time-dependent numerical simulations of reactive flows, so we introduce other approaches such as the diffusion approximations, flux approximations, and Monte Carlo methods.

For more detailed explanations and derivations, we recommend the books by Zeldovich and Raizer (1966), Özişik (1973), Rybicki and Lightman (1979), Mihalas and Mihalas (1984), and Kalkofen (1984) and the more recent books by Modest (1993) and the new edition of Siegel and Howell (1993). A review article by Viskanta and Mengüç (1987) presents a very readable description of the problems and advantages of many methods and gives a number of examples from practical engineering applications. The more recent review of Viskanta (1998) also provides an overview of the literature on coupled covection-radiation problems in combustion and heat transfer applications.

13–1. The Physical Basis of Radiation Transport

The radiation that is important in many reactive-flow problems has its origin in the interaction of photons and the medium through which they propagate. For example, this radiation may be caused by electronic, vibrational, and rotational transitions of the atoms and molecules of a gas, by molecular vibration in solids and liquids, by bound-electron transitions in solids, by photon-collision processes in plasmas, or by transformation among atomic or nuclear elements in nuclear reactions. Radiation is both absorbed and reemitted as it is transmitted through a medium. When an atom (or an ion) and a photon interact, the photon can be absorbed, leaving the atom in an excited state. Subsequently, the atom can relax to its original state or to some other state by emitting another photon. This relaxation may be spontaneous or may be stimulated by the passage of other photons. The different wavelength regimes of radiation are shown in Figure 13.1. Table 13.1 defines the symbols that are used in this chapter.

Electronic transitions are generally divided into three types: bound-bound transitions, bound-free transitions, and free-free transitions. In *bound-bound transitions (b-b)*, a photon interacting with an atomic system can cause a change from one bound level to another. For example, a photon can be absorbed so that the system is excited to a higher-energy state. A photon can also be emitted, and so the system then drops to a lower-energy state. Because atomic systems have discrete energy levels, b-b emission produces line spectra. These become band spectra when the system also has vibrational or rotational modes that broaden the transition. In *bound-free transitions (b-f)*, a photon interacting with a bound electron-atom system results in a system that has too much energy for the electron to stay bound, resulting in photoionization. An electron is emitted, and the extra energy becomes kinetic energy of the free electron. The reverse of this process, atomic capture of a free electron, results in the emission of a photon. These kinds of processes produce continuous absorption and emission spectra. The directions of the particles and photons before and after the transition are governed by momentum and energy conservation. In a *free-free transition (f-f)*, a free electron can either emit a photon without

Table 13.1. Symbols in Chapter 13

Symbol	Definition
$t, \rho, \rho\mathbf{v}, E, T, P, k_B, \lambda_m$	Symbols of quantities defined in Table 2.1
A	Surface area (cm^2)
c	Speed of light; in a vacuum, 2.998×10^{10} cm/s
c_{x_i}	Direction cosine for generic coordinate x_i
c_x, c_y, c_z	Direction cosines in Cartesian geometry
d	Dimension, $d = 1, 2, or\ 3$
$E_{\lambda b}$	Blackbody emissive power $= \pi I_{\lambda b}$ (erg s^{-1} cm^{-2})
E_b	Blackbody energy density (erg cm^{-3})
E_{bi}	Blackbody energy density from surface i (erg cm^{-3})
$\mathbf{E_b}$	Vector of $\{E_{bi}\}$ at surface zones i (erg cm^{-3})
E_r	Energy of the radiation field (ergs cm^{-3})
f_E	Variable Eddington factor
$\overline{g_i s_j}$	Surface-volume direct exchange area (cm^2)
$\overline{g_i g_j}$	Volume-volume direct exchange area (cm^2)
$\overline{G_i G_k}$	Gas-gas total exchange area (cm^2)
$\overline{G_i S_j}$	Gas-surface total exchange area (cm^2)
h	Planck's constant, 6.626×10^{-27} erg-s
$I(\lambda, \Omega)$, $I_\lambda(\Omega)$, or I_λ	Spectral intensity (erg s^{-1} cm^{-3} $ster^{-1}$)
$I(\Omega)$	Total intensity in direction (erg s^{-1} cm^{-2} $ster^{-1}$)
$\overline{I_\lambda}$	Mean intensity (erg s^{-1} cm^{-3})
$I_{\lambda b}$	Planck function, spectral intensity of blackbody radiation $= E_{\lambda b}$ (erg s^{-1} cm^{-3} $ster^{-1}$)
K	Number of volume zones in zonal model
K_p	Planck-averaged opacity (cm^{-1})
K_λ	Extinction coefficient (cm^{-1})
L	Radiation-field scale length (cm)
N	Number of surface zones in zonal model
Q	Heat flux at zone i (erg s^{-1})
$\mathbf{Q}$	Vector of $\{Q_i\}$ (erg s^{-1})
$\mathbf{q}_r$	Radiative heat flux (erg cm^{-2} s^{-1})
R	Reflectivity
s	Path of the radiation propagation (cm)
$S_{i,j}$	Path length between zones i and j (cm)
$\mathcal{S}$	Tensor containing direct exchange area (cm^2)
$\overline{s_i s_j}$	Surface-surface direct exchange area (cm^2)
$\overline{S_i S_j}$	Surface-surface total exchange area (cm^2)
T_r	Local radiation temperature (K)
V	Volume (cm^3)
W	Source term for RTE (erg s^{-1} cm^{-4} $ster^{-1}$)
α, α_λ	Absorption coefficient (cm^{-1})
α_R	Rosseland mean absorption coefficient (cm^{-1})

Symbol	Definition
ϵ	Emissivity
κ	Extinction coefficient (cm^{-1})
λ	Wavelength of radiation (cm)
λ_R	Rosseland mean free path (cm^{-1})
σ_B	Stefan-Boltzmann constant, 5.67×10^{-5} erg cm^{-2} K^{-4} s^{-1}
$\sigma_\lambda(s)$	Scattering coefficient (cm^{-1})
τ	Optical thickness
Φ	Scattering phase function
$\Omega = \Omega(\theta, \phi)$	Solid angle, direction of solid angle (ster)
Subscripts	
b	Blackbody
g	Gas
i, j, k	Indices over $i, j, k = 1, \ldots N$ or K
s	Surface
λ	Wavelength

losing all of its kinetic energy, and stay free, or it can absorb a photon and its kinetic energy increases. This is called *bremsstrahlung*, and produces continuous emission and absorption spectra.

These three basic atomic processes contribute to the absorption and emission of radiation as it passes through a material. The rates of these atomic radiation processes play much the same role for radiation transport that chemical reaction rates play for chemical kinetics. The radiation-rate data are the emission, absorption, and scattering coefficients. These constitute the fundamental input determining emission and absorption spectra, and are used in the equation of radiation transport given later. Determining these input quantities is a large part of the difficulty in solving radiation-transport problems, just as it is for complicated chemical kinetics mechanisms.

We are often interested in systems in *local thermal equilibrium*. This means that the radiation field can be described in terms of local state variables such as composition, temperature, and pressure. The photons may be absorbed and reemitted many times as they cross the material. When the radiation is in local thermal equilibrium, it is called *thermal radiation*.

Radiative-transfer theory is the quantitative study, on a macroscopic level, of the interaction of radiation with matter that absorbs, emits, and scatters radiant energy. It is derived from microscopic considerations and from conservation principles similar to those used to derive the Navier-Stokes equations. This theory considers rays of photons, and the macroscopic problem of how these rays are transformed as they pass through the medium. The detailed mechanism of the interaction process involving atoms or molecules and the radiation field is not considered. These detailed interactions are the subject of electromagnetics and quantum mechanics. Radiative-transfer theory, however, is quite appropriate for coupling to Navier-Stokes continuum representations of matter.

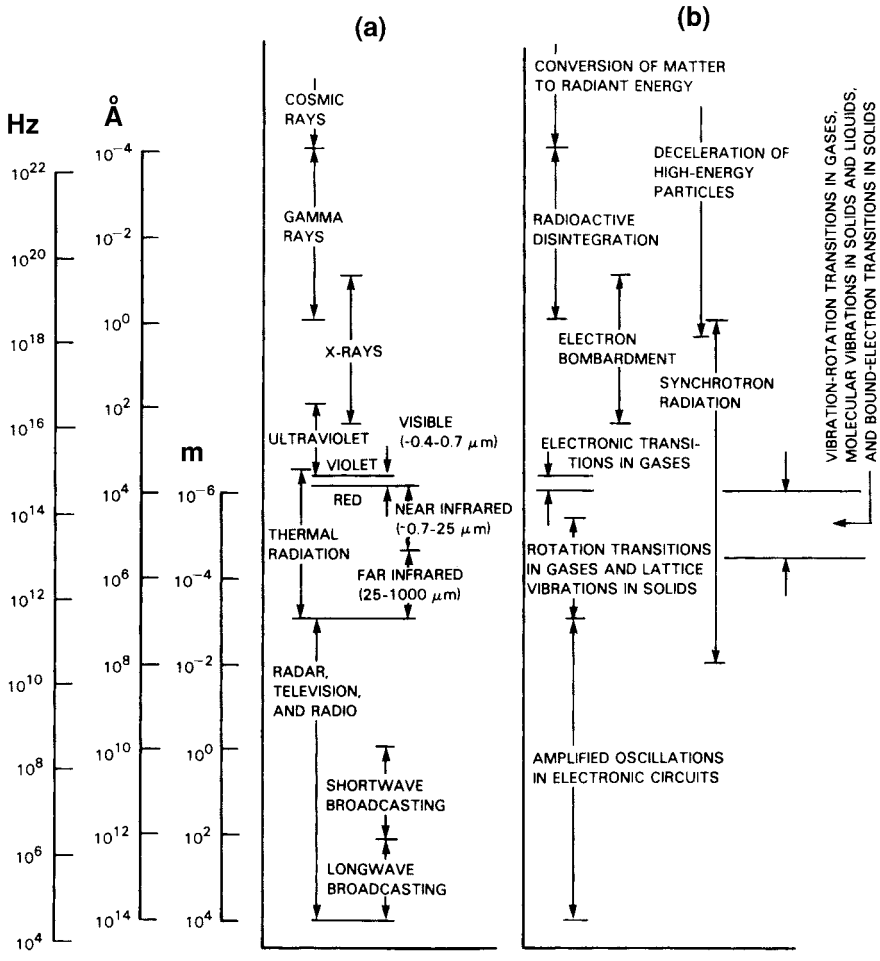

Figure 13.1. Spectrum of electromagnetic radiation. (a) Type of radiation. (b) Production mechanism. Other common units used are: wavelength in μm (1 μm $= 10^{-6}$m), and wavenumber in m^{-1} (1 m^{-1} $= 10^{8}$ Hz). (This figure, courtesy of R. Siegel, NASA-Glenn Research Center, was based on Siegel and Howell [1968].)

13–2. The Equations of Radiation Transport

13–2.1. The Radiative-Transfer Equation

The radiative-transfer equation (RTE) is a mathematical statement of conservation principles applied to a monochromatic beam of radiation. The equation has been derived from many starting points, including the Liouville and Boltzmann equations, quantum mechanics, and geometrical optics (see references, e.g., in Viskanta and Mengüç [1987]). The description given here is based on conservation principles that quantify the rate of change of radiation intensity along the path in terms of how the radiation is absorbed, emitted, and scattered. More complete derivations are given in the references.

The fundamental quantity that is computed by the RTE is the *spectral intensity* I_λ, the radiant energy passing through a surface, per unit time, per unit wavelength interval

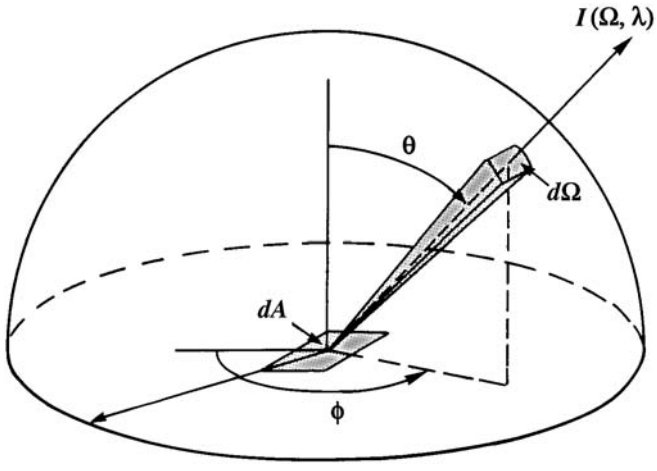

Figure 13.2. Geometry used to define the spectral intensity, $I(\lambda, \Omega)$.

about wavelength λ, per unit surface area normal to the $\Omega = (\theta, \phi)$ direction, into a unit solid angle centered around Ω. This is illustrated in Figure 13.2. Below we also refer to this quantity as $I_\lambda(\Omega)$, or simply I_λ, where the subscript λ indicates the wavelength dependence. The spectral intensity depends on material properties of the system, such as temperature, pressure, and the number densities of various species. To simplify the presentation, the explicit dependences on the local properties of the medium are dropped in the following discussion.

The *total intensity* in direction Ω is defined as

$$I(\Omega) \equiv \int_0^\infty I_\lambda(\Omega)\, d\lambda. \tag{13–2.1}$$

A direction-averaged *mean intensity* is defined as

$$\overline{I(\lambda)} = \frac{1}{4\pi} \int_0^{4\pi} I_\lambda(\Omega)\, d\Omega. \tag{13–2.2}$$

The intensity is a function of the global properties of the whole medium: how much radiation enters the medium from the walls or outside, how much is absorbed, emitted, or scattered in the medium before it reaches a small volume, and how much is absorbed and emitted in that volume.

The equation of radiative transfer describes the behavior of the spectral intensity I_λ. In general it is a time-dependent conservation equation,

$$\frac{1}{c}\frac{\partial I_\lambda}{\partial t} + \frac{\partial I_\lambda}{\partial s} = W_\lambda, \tag{13–2.3}$$

where the independent variable s measures distance along a ray in direction Ω, and W_λ is a combination of source terms,

$$W_\lambda \equiv W_{\text{emission}} - W_{\text{absorption}} + W_{\text{scattering in}} - W_{\text{scattering out}}, \tag{13–2.4}$$

that represents photon absorption, emission, and scattering in and out of Ω. We usually

require only the steady solution of equation (13–2.3). Radiation, with c as the characteristic velocity, relaxes toward equilibrium faster than any other process. The radiative time scale for problems involving coupled radiation transport and fluid dynamics is generally much less than the relevant fluid time scales, except for very high-energy problems. The radiation field adjusts itself quickly to changes in the physical conditions at any position in the medium. Thus it is reasonable to ignore explicit time variations in $I_\lambda(\Omega)$ and treat the radiation field as a sequence of quasisteady states.

The decrease in I_λ due to absorption of photons along the *ray* parameterized by s, is given by

$$W_{\text{absorption}} = \alpha_\lambda(s) I_\lambda(s), \tag{13–2.5}$$

where $\alpha_\lambda(s)$ is the absorption coefficient. As written in equation (13–2.5), $\alpha_\lambda(s)$ contains both absorption and induced emission, each proportional to the spectral intensity in the ray. The total absorption may be positive or negative. When induced emission dominates, α_λ is a positive gain coefficient describing the growth of the spectral intensity along the ray.

The increase in I_λ due to spontaneous emission in the material along the paths is

$$W_{\text{emission}} = \alpha_\lambda(s) I_{\lambda b}(s). \tag{13–2.6}$$

If we assume that the radiation is in local thermodynamic equilibrium in equation (13–2.6), $I_{\lambda b}$ is the *Planck function*,

$$I_{\lambda b}(T) \equiv \frac{2hc^2}{\lambda^5 \left(\exp(hc/k_B \lambda T) - 1\right)}, \tag{13–2.7}$$

where h is Planck's constant. The function $I_{\lambda b}$ describes the spectral intensity of *blackbody radiation*, the intensity found in an enclosed vacuum whose walls are at a uniform temperature. The Stefan-Boltzmann law states that the blackbody intensity integrated over all wavelengths is proportional to T^4, that is,

$$I_b(\Omega) = \int_0^\infty I_{\lambda b}(\Omega) d\lambda = \frac{\sigma_B}{\pi} T^4, \tag{13–2.8}$$

where σ_B is the Stefan-Boltzmann constant.

The attenuation by scattering is

$$W_{\text{scattering out}} = \sigma_\lambda(s) I_\lambda(s), \tag{13–2.9}$$

where σ, the scattering coefficient, is a function of wavelength λ and evaluated along the line s. The radiation scattered out of a volume is proportional to the radiation present, as is absorption. The scattering coefficient σ_λ, a function of $T(s)$ and $P(s)$, is the reciprocal of the mean free path of a photon of wavelength λ along the path s. The scattering coefficient describes elastic and inelastic scattering and may be isotropic or anisotropic in the directions into which the ray scatters.

The gain of intensity by radiation scattered into the ray along s is

$$W_{\text{scattering in}} = \frac{\sigma_\lambda}{4\pi} \int I_\lambda(s, \Omega_i)\Phi(\lambda, \Omega, \Omega_i)\, d\Omega_i \ , \tag{13–2.10}$$

where Φ is the *phase function*, defined as the scattered intensity in one direction, divided by the intensity for isotropic scattering.

Therefore, the total equation of radiative transfer in the quasistatic limit is a first-order integro-differential equation,

$$\frac{dI_\lambda}{ds} = -\alpha_\lambda I_\lambda(s) - \sigma_\lambda I_\lambda(s) + \alpha_\lambda I_{\lambda b}(s) + \frac{\sigma_\lambda}{4\pi} \int_0^{4\pi} I_\lambda(s, \Omega_i)\Phi(\lambda, \Omega, \Omega_i)\, d\Omega_i . \tag{13–2.11}$$

This equation describes how the radiation along a path s is attenuated by absorption and scattering, and is enhanced by spontaneous emission, induced emission (included in the net absorption coefficient α_λ), and radiation scattering into the path from other directions.

Both σ_λ and α_λ are functions of wavelength, the chemical species present, the temperature, and the pressure, but not, in general, the photon direction. Together, they define the *extinction coefficient*, $\kappa_\lambda(s)$, where

$$\kappa_\lambda \equiv \alpha_\lambda + \sigma_\lambda . \tag{13–2.12}$$

The *optical thickness* or *opacity* of a slab of material, $\tau_\lambda(s)$, is a measure of the ability of a given path length s to attenuate radiation,

$$\tau_\lambda(s) \equiv \int_0^s \kappa_\lambda(s')\, ds' . \tag{13–2.13}$$

When $\tau_\lambda(s) \gg 1$, the material is *optically thick*. This means that the mean penetration distance of each photon is small compared to the characteristic dimensions in the medium. When $\tau_\lambda(s) \ll 1$, the material is *optically thin*. When it is assumed that the radiative properties such as α and σ, and therefore κ are independent of λ, the medium is called *gray*.

The accuracy of any solution of the RTE is based on the same sorts of considerations as those of the Navier-Stokes equations. For a specific problem, we must consider accuracies in the input data, any theoretical simplifications made to represent the problem, and inaccuracies in the numerical solution method applied to the model. As with much of the input data for the reactive Navier-Stokes equations, the coefficients in a radiation-transport problem are often poorly known. For example, a combustion process may produce water vapor, carbon dioxide, and carbon monoxide. These gases do not scatter radiation significantly, but they can absorb or emit radiation strongly. The presence of particulates, such as soot or ash, also has a strong effect on the radiative properties of a system. When particulates are present, it might be necessary to consider their size and spatial distribution. Exactly how the material and spectra should be considered varies considerably for each type of system and for the accuracy required of the solution.

13–2.2. The Radiant Energy Flux

In Chapter 2, we introduced the radiant energy flux $\mathbf{q}_r$ which, through the term $\nabla \cdot \mathbf{q}_r$, couples the radiation transport to the other chemical and physical processes in the conservation equations. The radiant energy flux crossing an area element dA is a result of the intensities from all directions,

$$\mathbf{q}_r \equiv \int_0^\infty \int_0^{4\pi} I_\lambda(\Omega, s) \cos\theta \, d\Omega \, d\lambda, \tag{13–2.14}$$

where θ is the angle between each ray direction Ω and the surface normal to dA. To evaluate $\nabla \cdot \mathbf{q_r}$, we need to evaluate the gradient of I_λ along s, in the appropriate coordinate system for the computation,

$$\frac{dI_\lambda}{ds} = \frac{\partial I_\lambda}{\partial x_1}\frac{dx_1}{ds} + \frac{\partial I_\lambda}{\partial x_2}\frac{dx_2}{ds} + \frac{\partial I_\lambda}{\partial x_3}\frac{dx_3}{ds} = \frac{\partial I_\lambda}{\partial x_1}c_{x_1} + \frac{\partial I_\lambda}{\partial x_2}c_{x_2} + \frac{\partial I_\lambda}{\partial x_3}c_{x_3}, \tag{13–2.15}$$

where the $\{c_{x_i}\}$ are the direction cosines of Ω for the coordinate system $\{x_i\}$. For example, for three-dimensional Cartesian coordinates (x, y, z),

$$\frac{dI_\lambda}{ds} = \frac{\partial I_\lambda}{\partial x}c_x + \frac{\partial I_\lambda}{\partial y}c_y + \frac{\partial I_\lambda}{\partial z}c_z, \tag{13–2.16}$$

where

$$\begin{aligned} c_x &= \sin\theta\cos\phi \\ c_y &= \sin\theta\sin\phi \\ c_z &= \cos\theta. \end{aligned} \tag{13–2.17}$$

Then by integrating equation (13–2.11) to find I_λ, and using this in equation (13–2.14), we obtain

$$\nabla \cdot \mathbf{q}_r = 4\pi \int_0^\infty \left(\alpha_\lambda(s) I_{\lambda b} - \kappa_\lambda \overline{I_\lambda} + \frac{\sigma_\lambda(s)}{4\pi} \int_0^{4\pi} I_\lambda(s, \Omega_i)\, \overline{\Phi}(\lambda, \Omega_i)\, d\Omega_i \right) d\lambda, \tag{13–2.18}$$

where

$$\overline{\Phi}(\lambda, \Omega) = \frac{\sigma_\lambda(\Omega)}{\langle\sigma\rangle} \tag{13–2.19}$$

and

$$\langle\sigma\rangle \equiv \frac{1}{4\pi}\int_0^{4\pi} \sigma_\lambda(\Omega)\, d\Omega. \tag{13–2.20}$$

In equation (13–2.18), $\overline{I_\lambda}$ is the mean intensity defined above in equation (13–2.2).

Equations (13–2.11) and (13–2.18) look complicated but straightforward to solve. Nonetheless, there are two significant types of difficulties inherent in solving them. The

first is determining the absorption and scattering coefficients, which require a detailed knowledge of the atomic and molecular properties of a medium. The second is the enormous amount of computation required to solve the equations numerically because each direction must be considered separately. Even assuming local thermodynamic equilibrium, there is also the problem of solving for each I_λ, which couples to the equations for energy and temperature.

Despite these complications, there are a number of exact solutions of the equations, given certain simplifying assumptions about the input data, the nature of the radiation field, and the geometry. These solutions are analogous to the exact solutions of the Navier-Stokes equations. In both cases, exact solutions provide extremely valuable information for benchmarking numerical solutions and represent limiting cases that are important for constructing algorithms. Analytic solutions of the RTE generally assume that the radiative properties are uniform and the boundary conditions are homogeneous. For example, there has been extensive development of one-dimensional solutions for neutron transport, heat transfer, and atmospheric problems. There are also some three-dimensional analytic solutions, again with simplifying assumptions about the boundary conditions, geometry, and radiative properties. These exact solutions are discussed and reviewed in some detail by Siegel and Howell (1993) and Özişik (1973), and a good summary of the work has been given by Viskanta and Mengüç (1987).

Because of the mathematical and computational difficulties of solving general integro-differential equations, there has been extensive development of approximate methods for solving the RTE. Some of these are based on assumptions about the opacity of the material, and some involve assumptions about the angular distribution of $I(\Omega)$. Others are based on more formal mathematical truncations of series expansions that lead to higher-order aproximations. In the material that follows, we review some of the more commonly used methods and indicate their advantages and limitations.

13–2.3. Important Limiting Cases

First consider the integral of dI_λ/ds from equation (13–2.11) to determine $I_\lambda(s)$,

$$I_\lambda(s) = I_\lambda(0)e^{-\int_0^s \kappa_\lambda ds'} + \int_0^s \kappa_\lambda I_\lambda(s')e^{-\int_{s'}^s \kappa_\lambda ds''} ds', \tag{13–2.21}$$

where κ_λ, defined in equation (13–2.12), can vary along the path. By comparing this to equation (13–2.11), we see that $I_\lambda(s)$ depends on the Planck function $I_{\lambda b}$ and thus on the local temperature and on the local scattered radiation.

A practical computational approach must avoid solving the full radiative-transport equations, with both spatial and wavelength dependencies, coupled to the Navier-Stokes equations through $\nabla \cdot \mathbf{q}_r$ in the energy equation. Instead, it is necessary to neglect terms, exploit the limiting cases of thick or thin radiation, and use an assortment of approximations to simplify the equations. First consider some of the limiting cases.

The *transparent gas* approximation assumes that the medium is optically thin, and that the local radiation intensity is dominated by the radiation energy incident at the boundaries. The exponential attenuation terms in equation (13–2.21) approach unity (no attenuation),

and we find that

$$I_\lambda(s) = I_\lambda(0) + \int_0^s \kappa_\lambda I_\lambda(s')\, ds'. \tag{13–2.22}$$

The attenuation along the path s is very small, so that radiation enters at $s = 0$ and maintains essentially the same intensity. If the extinction coefficient κ_λ is small enough that the integral in equation (13–2.22) is small compared to $I_\lambda(0)$,

$$I_\lambda(s) = I_\lambda(0), \tag{13–2.23}$$

which is the *strongly transparent approximation*. Then the incident intensity is not changed as the radiation travels through the medium. A local energy balance is appropriate, and it is much easier to solve than the full transport equations.

In the *emission approximation* limit, the medium is optically thin, and not much radiation is entering from the boundaries. Therefore, the spontaneously emitted energy from each small volume of the gas passes through and out of the system without much attenuation. Thus the only emission is from the medium itself. Then $I_\lambda(0)$ is negligible, and

$$I_\lambda(s) = \int_0^s \alpha_\lambda(s') I_{\lambda b}(s')\, ds'. \tag{13–2.24}$$

In another limit, the radiation emitted and scattered by the medium is small compared to that incident from the boundaries or external sources. The local intensity is then the attenuated incident intensity,

$$I_\lambda(s) = I_\lambda(0) e^{-\int_0^s \kappa_\lambda(s')\, ds'}. \tag{13–2.25}$$

This limit could be important in combustion problems when a cold unignited gas mixture with significant absorbers such as smoke particles is adjacent to a region of strong emission.

Another type of limit occurs when the optical depth of the medium is large enough and the temperature gradients are small enough. Then the local radiation intensity results only primarily from local emission. This gives rise to the various types of diffusion approximations, some of which are discussed in Section 13–3.

13–2.4. Monte Carlo Methods

Monte Carlo methods are not based on a single technique, but are a class of rather loosely related techniques. The concept that defines this class is the use of statistical sampling to determine the likely outcome of an interaction. The first use of the Monte Carlo approach relevant to the discussion here was by Metropolis and Ulam (1949). Monte Carlo methods were used originally for neutron transport in fission materials and have since been applied extensively to astrophysical and atmospheric problems. To a lesser extent, they have also been used for computing radiative-heat transfer in engineering systems. Good general references and introductions can be found in Hammersley and Handscomb 1964), Fleck (1961), Cashwell and Everett (1959), Schreider (1964), Brown (1970), and Howell (1968). A good recent exposition on applying Monte Carlo

techniques to heat transfer for engineering problems can be found in Siegel and Howell (1993) and Modest (1993).

Applying a Monte Carlo method to radiation transport means tracing the history of a statistically meaningful random sample of photons from points of emission to the points of absorption. The method consists of simulating a finite number of representative photon histories. For each photon, random numbers are generated and used to sample appropriate probability distributions for scattering angles and path lengths between collisions. A computation is started by assigning a set of values to the photon, such as its initial energy, position, and direction. The number of mean free paths that the photon propagates is determined stochastically. Then, the cross-section data are sampled and these are used to determine whether the collided photon is absorbed or scattered by the gas molecules or particles in the medium. If the photon is absorbed, the computation for that photon is terminated. If the photon is scattered, the distribution of scattering angles is sampled and a new direction is assigned to the photon, depending on whether the scattering is elastic or inelastic. For elastic scattering, a new energy is determined by conservation of energy and momentum. With the newly assigned energy, position, and direction, the procedure is repeated for successive collisions until the photon is absorbed or escapes from the system.

Monte Carlo solutions fluctuate statistically around the "real" answer. A property of statistical methods is that the solutions improve in accuracy as these fluctuations decrease. The answer is expected to converge to the exact solution of the problem as the number of photons increases. The beauty of the approach is that it can correctly solve problems with complex physics and geometry with relatively uncomplicated programming. The problem with the method is fundamentally the computational cost of using enough photons to obtain meaningful answers. Monte Carlo methods are most accurate when the opacity is small (the medium is optically thin), and it may descrease drastically for moderate-to-large opacities. This is in direct analogy to the problems in using DSMC that were discussed in Chapter 9. In that case, as the mean free paths of the participating molecules (not photons) become reasonably small compared to the system size, statistical fluctuations dominate and the cost of obtaining a good solution goes up dramatically. In both cases, when the optical thickness is too small, or the Knudsen number is too low, it is more economical to use another method.

13–3. Radiation Diffusion Approximations

Radiation diffusion approximations are the easiest models to use. They are valid when the radiation field is approximately isotropic locally. The combination of photon paths followed by absorption and reemission results in a random-walk behavior whose macroscopic representation is, to lowest order, a diffusion operator. The diffusion approximation is based on the assumption that the optical depth is large enough (the mean free path of the photons is small, or the medium is optically thick) and the temperature gradients are small enough for the local value of the spectral intensity to be a function of local emission only.

There are many variations of the diffusion approximation, and these are described in detail in a number of texts (see, e.g., Siegel and Howell [1993]). For example, the

Schuster-Schwarzschild approximation (Schuster 1905; Schwarzschild 1906) and the Milne-Eddington approximation (Milne 1930; Eddington 1959) are both diffusionlike approximations. Both of these assume that radiation propagates in only two directions, and that the intensity in one direction may have a different value than the intensity in the other. They differ, however, in exactly when in the theory they make the two-direction assumption. It is possible within this framework to consider the behavior of different wavelength bands when it is valid to assume isotropy in each of the bands.

13–3.1. The Standard Radiation Diffusion Approximation

An expression for the radiation energy flux, first derived by Rosseland (1936), is

$$\mathbf{q}_r = -\frac{4\sigma_B}{3\alpha_R}\nabla T^4, \tag{13–3.1}$$

which comes from relating the energy flux to the local material temperature gradient. This expression gives the diffusive flux of radiation averaged over all wavelengths assuming a local blackbody distribution.

The quantity α_R is the *Rosseland mean absorption coefficient*, an average over the absorption and scattering coefficients α_λ and σ_λ weighted by the blackbody distribution function $I_{\lambda b}(T)$,

$$\alpha_R \equiv \frac{\displaystyle\int_0^\infty \frac{\partial I_{\lambda b}(T)}{\partial T}\, d\lambda}{\displaystyle\int_0^\infty \frac{\partial I_{\lambda b}(T)}{\partial T}\frac{d\lambda}{\alpha_\lambda + \sigma_\lambda}}. \tag{13–3.2}$$

The mean free path of photons corresponding to α_R is the Rosseland mean free path,

$$\lambda_R \equiv \frac{1}{\alpha_R}. \tag{13–3.3}$$

Once $\mathbf{q}_r$ is determined, the energy conservation equation can be solved by standard numerical methods described in previous chapters.

13–3.2. The Variable Eddington Approximation

The variable Eddington approximation is a diffusion model that considers small, linear anisotropies and two directions of integration. Treating many, distinct wavelength bands in the radiation field is possible within the framework of this model, when it is valid to assume isotropy in each of the bands.

Consider a situation in which the local radiation temperature T_r deviates significantly from the local material temperature T. In such cases, both the photons and the material may be described essentially as Maxwellian temperature distributions, but at different temperatures. Since the photons and the medium do interact, there must be some coupling, however weak, that tends to equilibrate the temperatures. In this case, a computationally useful model results by treating the radiation field in terms of a single energy density E_r

related to T_r by

$$T_r \equiv \left(\frac{cE_r}{4\sigma_B}\right)^{1/4}. \tag{13–3.4}$$

The total energy in the system still has to be conserved, but the fluid energy E and the radiation energy E_r interchange locally.

The basic equation for implementing the variable Eddington approximation is an energy-transport equation for radiation,

$$\frac{1}{c}\frac{\partial E_r}{\partial t} = \nabla \cdot \lambda_R \nabla(f_E E_r) - K_p(E_r - E_b), \tag{13–3.5}$$

where scattering is treated in a gradient diffusion approximation using the Rosseland mean free path λ_R. The quantity f_E is the dimensionless variable Eddington factor discussed below. The absorption and emission are governed by the *Planck-averaged opacity*, K_p,

$$K_p \equiv \frac{\int_0^\infty \kappa_\lambda I_{\lambda b}(T)\, d\lambda}{\int_0^\infty I_{\lambda b}(T)\, d\lambda}, \tag{13–3.6}$$

the wavelength-averaged extinction coefficient. The opacity depends on the composition and temperature of the fluid and, more generally, on the wavelength of the photons.

The blackbody energy density appearing in equation (13–3.5) is defined as

$$E_b \equiv 4\frac{\sigma_B T^4}{c}. \tag{13–3.7}$$

When the actual radiation energy density exceeds the blackbody radiation density, E_b, evaluated at the fluid temperature T locally, the radiation energy density decreases in time because the fluid absorbs radiation faster than it is being emitted.

The left side of equation (13–3.5) is the rate of change of the energy density of the radiation, generally a very small quantity relative to the energy density of the fluid. It is also generally negligible in equation (13–3.5) because the velocity of light is large compared to other speeds in the problem. The variable Eddington factor, f_E, ranges between one for optically thin media and one-third for optically thick media, as shown schematically in Figure 13.3. This factor gives the model its ability to approximate the local radiation flux in regions bridging the optically thick and optically thin conditions. Generally, the variable Eddington factor should be computed from a detailed solution of the equations of radiative transfer. For computational simplicity, it is often approximated by a formula based on the geometry of the problem and the ratio of the radiation flux to the radiation energy density.

One example is

$$f_E = \frac{L/3 + \lambda_R}{L + \lambda_R}, \tag{13–3.8}$$

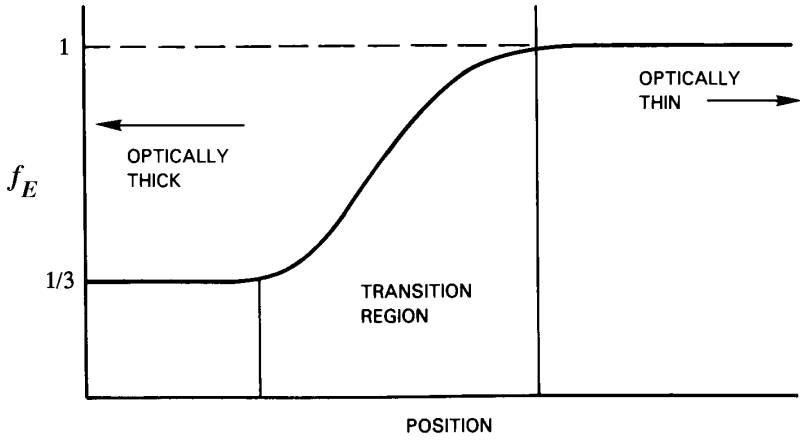

Figure 13.3. Variation of the variable Eddington factor, f_E, from an optically thick to an optically thin region.

where L is the radiation field scale length determined from

$$L^{-1} \approx \frac{f_E |\nabla E_r|}{E_r}. \tag{13–3.9}$$

Since f_E enters the definition of L, a quadratic equation has to be solved to unfold equations (13–3.8) and (13–3.9). This is the *ad hoc* part of the approximation. The variable Eddington factor, f_E, could also be taken inside the gradient term in equation (13–3.9), as used in equation (13–3.5). When f_E is taken inside the gradient, the radiation field scale length has to be determined using the values of f_E and ∇E_r from the previous timestep.

When the radiation field has relaxed to a steady state, the diffusionlike term in equation (13–3.5) balances the radiation source term, and so the left side of equation (13–3.5) vanishes. The radiation field is tightly coupled to the boundary conditions and to the spatial variation of the fluid temperature field through the blackbody energy density. This elliptic equation can be solved by standard techniques as discussed in Chapter 10, although doing so is expensive in multidimensions.

The fluid dynamic energy density satisfies a similar equation,

$$\frac{\partial E}{\partial t} + \nabla \cdot E\mathbf{v} + \nabla \cdot P\mathbf{v} - \nabla \cdot \lambda_m \nabla T = cK_p(E_r - E_b), \tag{13–3.10}$$

where λ_m is the thermal-conduction coefficient of the material, described and used in Chapters 2 and 7. The velocity of light now occurs in the numerator of the emission and absorption term. This means that even though the radiation energy density is much smaller than the fluid energy density, the fluid energy can be changed appreciably by the radiation because the speed of light is large. This also means that the difference $(E_r - E_b)$ must be accurately determined and implemented exactly as is used in equation (13–3.5).

This coupled system of two energy equations conserves global energy, as we can see by multiplying equation (13–3.5) by c and adding it to equation (13–3.10). Also, the variation of fluid energy density, and hence temperature, feeds back into equation (13–3.5) through the temperature variation of the blackbody energy density $E_b(T)$ and the Rosseland mean

$\lambda_R(T)$. Therefore the system can be quite stiff. This potential stiffness of the coupled equations complicates their numerical solution, but the techniques described in Chapter 11 are adequate, at least in principle, to deal with this.

The variable Eddington approximation reduces to the diffusion model when E_b and E_r are nearly equal. In this case, the elliptic term in equation (13–3.5) can be written with $E_b(T)$ in place of E_r. When the result is substituted for the right side of equation (13–3.10), we find

$$\frac{\partial E}{\partial t} + \nabla \cdot E\mathbf{v} + \nabla \cdot P\mathbf{v} - \nabla \cdot \lambda_m \nabla T = \nabla \cdot \lambda_R \nabla \left(\frac{4}{3} \sigma_B T^4 \right). \tag{13–3.11}$$

By comparing equation (13–3.11) with the Rosseland diffusion flux given in equation (13–3.1), we can see that the variable Eddington model reduces to the diffusion model in the optically thick limit, $f_E = 1/3$.

In the optically thin limit, it is sometimes necessary to put a limit on the radiation flux. The limit ensures that the amount of radiation flux is bounded by its free-streaming value, $c\,E_r$. Physically the flux cannot exceed this value, but mathematically it can in a diffusion model, as discussed in Chapter 7. The breakdown in this case comes about because the system may enter a regime where the diffusion approximation cannot be valid.

The computational problem becomes difficult when the spectral intensity varies markedly with direction. This occurs, for example, where a medium undergoes a transition from optically thick to optically thin. At the boundary of a dense radiating region, for example, the radiation distribution changes from isotropic to essentially unidirectional streaming away from the source. Determining an accurate description of this phenomenon is a challenge. Very often the variable Eddington method is fully consistent with the level of approximation of the problem. When it is not, the geometric complexity again requires a full nonlocal treatment of the radiation.

One important application of the variable-Eddington method is in the FAST program, which is used to simulate laser-matter interactions in the context of inertial confinement fusion (Gardner et al. 1998). The computer program couples models of fluid dynamics, thermal conductivity, and complex and variable equations of state to a multigroup model for radiation transport. The radiation is assumed in steady state, as the radiative relaxation is very short compared to relevant fluid time scales. The multigroup model (Dahlburg et al. 1995) results in an additional elliptic equation for each frequency group in the spectrum. This is a production model in the sense that it has undergone rather elaborate tests of its submodels, algorithms, and input data. The variable-Eddington method has also been used as part of a reactive-flow model that describes confined combustion flows over bluff bodies (Fureby and Möller 1995).

13–4. Other Solution Methods

The work involved in a direct, brute-force solution of the multidimensional equations of radiative transfer is straightforward, but horrendous. First, there is the considerable problem of determining the appropriate coefficients for the various emission and absorption processes. These are usually a function of wavelength, as well as of the local properties of the material, and may also require information about the line radiation. Sometimes the

coefficients are available in tabular form, but often they must be extrapolated or estimated from very basic considerations. The spectrum is usually divided into wavelength bands and the radiative-transport equations are recast as a collection of equations, each extending over a wavelength band of width $\Delta\lambda$. Furthermore, space is divided into a finite number of directions so that photons traveling in different directions are treated separately. The general result is a very complex set of coupled finite-difference equations that nominally can be solved by standard methods described in previous chapters of this book. Implicit methods are usually best because of the nonlocal coupling involved.

There are approaches, other than those described in Section 13–3, that represent more fundamental models and incorporate more of the features of exact solutions. These approaches are computationally more intensive, but are sometimes required for problems in which radiation-transport effects must be considered with greater accuracy. For example, in many radiating systems, the preferential absorption of some frequencies means photons from other spectral bands have very long mean free paths and hence a better chance to escape. In other systems, radiation cooling leaves certain atoms at a different effective temperature than others. In these complex situations, different parts of the spectrum have to be treated differently, and some form of multidirectional, multigroup treatment becomes necessary. Now we summarize some of the features of these more complex methods.

13–4.1. The Zonal Method

The *zone method*, also called the *zonal method* or *Hottel's zonal method* (Hottel and Cohen 1958; Hottel and Sarofim 1967; Noble 1975), is one of the most commonly used methods for practical engineering problems. It is used for evaluating radiation heat transport in absorbing, emitting, and isotropically scattering media as occur in furnaces and other closed combustion systems (see, e.g., Khalil [1982], and references therein). Overviews of this method are given in many articles and textbooks that describe methods for radiation transport. A clearly presented, systematic exposition is given in the text by Modest (1993), and the discussion given below follows his approach.

The first step in the zonal method is to divide an enclosed area into zones, which consist of surface zones bounding the space and volume zones inside the space. A fundamental assumption is that the zones are small enough that it is reasonable to assume that the temperature in each zone is constant, at least for the treatment of radiation. Then through a series of assumptions about the absorption, emission, and scattering properties of the surface and gas medium, the energy balances represented by the RTE are reduced to a set of nonlinear algebraic equations for the unknown temperatures and the heat fluxes from the zones.

Perhaps the best way to get the idea of how this works is first to consider the simplest case, and then move on to several extensions. The simplest case is one in which there is no "participating medium," that is, nothing that absorbs or scatters or reflects in the volume, and the enclosing surfaces are black. The surface is divided into N isothermal zones. In this case, the net exchange of radiation energy Q between two surface zones i and j is

$$Q_{i,j} = -Q_{j,i} = \overline{s_i s_j}(E_{bi} - E_{bj}), \quad i, j = 1, 2, \ldots, N, \tag{13–4.1}$$

where E_{bi} is the blackbody radiation emitted from surface i. The *direct exchange areas*

$\overline{s_i s_j}$ are geometrical factors with units of area,

$$\overline{s_i s_j} = \overline{s_j s_i} = \int_{A_i}\int_{A_j} \frac{\cos\theta_i \cos\theta_j}{\pi S_{i,j}^2}\, dA_j dA_i, \tag{13–4.2}$$

where θ_i and θ_j are defined from the surface areas i and j, and $S_{i,j}$ is the path length between i and j. The net heat flux at zone i is then

$$Q_i = A_i q_i = \sum_{j=1}^{N} \overline{s_i s_j}(E_{bi} - E_{bj}) = A_i E_{bi} - \sum_{j=1}^{N} \overline{s_i s_j} E_{bj}, \qquad i = 1, 2, \ldots, N, \tag{13–4.3}$$

since $\sum_{j=1}^{N} \overline{s_i s_j} = A_i$. This equation can be written in matrix form as

$$\mathbf{Q} = \mathcal{S} \cdot \mathbf{E_b}, \tag{13–4.4}$$

where the elements of the $N \times N$ matrix $\mathcal{S}$ is composed of the direct exchange areas $\overline{s_j s_i}$. This is the simplest form of the zonal model. For any given enclosure, $\overline{s_i s_j}$ can be evaluated ahead of time and stored for use in the computation.

The next level of difficulty is to replace the blackbody surfaces with gray diffuse surfaces, so that the surfaces are partially reflective. The energy exchange between the surface zones now includes contributions due to reflections from other surface zones. The effect is that equation (13–4.1) must be changed appropriately, and we can now write

$$Q_{i,j} = -Q_{j,i} = \overline{S_i S_j}(E_{bi} - E_{bj}), \qquad i, j = 1, 2, \ldots, N, \tag{13–4.5}$$

where the $\overline{S_i S_j}$ are now called the *total exchange areas* that include the energy exchange by radiation traveling directly between i and j, as well as from the paths that involved multiple reflections from surfaces before they arrived along the path between i and j. Then equation (13–4.5) can be summed over j to obtain

$$Q_i = A_i q_i = \sum_{j=1}^{N} \overline{S_i S_j}(E_{bi} - E_{bj}) = \epsilon_i A_i E_{bi} - \sum_{j=1}^{N} \overline{S_i S_j} E_{bj}, \qquad i = 1, 2, \ldots, N. \tag{13–4.6}$$

The emissivity, ϵ_i, of zone i is defined as the ratio of the energy emitted from zone i to the energy emitted by a black surface at the same temperature. Generally,

$$\epsilon(T, \lambda) \equiv \frac{I(T, \lambda)}{I_{\lambda b}}. \tag{13–4.7}$$

The next issue is to relate the $\{\overline{S_i S_j}\}$ to the $\{\overline{s_i s_j}\}$, and there are many ways of expressing this. Modest (1993) uses a heat-balance equation and considerable matrix manipulation to determine that

$$\overline{\mathcal{S}\mathcal{S}} = \mathbf{T}^{-1} \cdot \mathcal{S} \tag{13–4.8}$$

where $\overline{\mathcal{S}S}$ is the matrix of $\overline{S_i S_j}$'s and the $\mathbf{T}$ and $\mathcal{S}$ are matrices that are functions of $\{\overline{s_i s_j}\}$, $\{\epsilon_i\}$, $\{A_i\}$, $\{R_i\}$, where R_i is the reflectivity, defined as the ratio of that part of the incoming radiation that undergoes reflection to the total incoming radiation.

Now the development can be complicated further by considering the effects of an absorbing and emitting material. Consider first a gray, absorbing, emitting, but nonscattering medium for which the absorption coefficient α is constant (not a function of local properties such as T or P). This medium is enclosed by diffusely emitting and reflecting gray surfaces. In addition to the N isothermal surface zones, the medium is broken into K isothermal volume zones. Now in addition to surface-surface direct exchange areas, there are surface-volume and volume-volume direct exchange areas. These new exchange areas are written as $\overline{g_i s_j}$ and $\overline{g_i g_j}$, respectively. The surface-surface areas must be modified to account for the attenuation of radiation due to the gas,

$$\overline{s_i s_j} = \int_{A_i} \int_{A_j} e^{-\kappa S_{i,j}} \frac{\cos\theta_i \cos\theta_j}{\pi S_{i,j}^2} \, dA_j \, dA_i, \tag{13–4.9}$$

where $e^{-\kappa S}$ is the transmission factor. Similarly,

$$\overline{g_i s_j} = \int_{V_i} \int_{A_j} e^{-\kappa S_{i,j}} \frac{\cos\theta_j}{\pi S_{i,j}^2} \kappa \, dA_j \, dV_i \tag{13–4.10}$$

$$\overline{g_i g_j} = \int_{V_i} \int_{V_j} e^{-\kappa S_{i,j}} \frac{\kappa^2}{\pi S_{i,j}^2} \, dV_j \, dV_i. \tag{13–4.11}$$

Then a detailed energy balance leads to the equations of the form

$$Q_{si} = \epsilon_i A_i E_{bsi} - \sum_{j=1}^{N} \overline{S_i S_j} E_{bsj} - \sum_{k=1}^{K} \overline{S_i G_k} E_{bgk}, \quad i = 1, 2, \ldots, N, \tag{13–4.12}$$

and

$$Q_{gi} = 4\kappa V_i E_{bgi} - \sum_{j=1}^{N} \overline{G_i S_j} E_{bsj} - \sum_{k=1}^{K} \overline{G_i G_k} E_{bgk}, \quad i = 1, 2, \ldots, K, \tag{13–4.13}$$

which now use total exchange areas. The same type of matrix manipulations, only somewhat more complex, may be used to express these equations in terms of the direct exchange areas. The derivations become even more complicated when the media is scattering or nongray.

The accuracy of the zonal method may be improved by either increasing the number of zones, increasing the accuracy of the numerical quadrature used to determine individual exchange areas, or increasing the accuracy of the input data. Increasing the number of zones allows the use of simple efficient methods to evaluate exchange areas, but requires time-consuming inversion of a larger matrix. The zonal method cannot be easily adapted for complicated geometries because many more exchange factors between zones must be evaluated and stored in computer memory. A good way to find exchange factors is by a Monte Carlo calculation.

13–4.2. Flux or Expansion Methods

The radiation intensity I is a function of location (x, y, z), direction of propagation (or angular dependence) $(\Omega(\theta, \phi))$, and the wavelength (λ). One of the major complications of solving the RTE is the existence of the angular dependence of I, which means that all directions have to be considered. The complete solution, then, is five-dimensional rather than three-dimensional. It would greatly simplify the solutions to separate the angular and spatial dependence in some way. This idea is the basis of the class of flux methods, which includes the multiflux, moment, discrete-ordinate, and spherical-harmonic methods. To the lowest order, many of these methods can be shown to be equivalent to each other. Krook (1955) has shown that the moment method, the spherical-harmonic method, and the discrete-ordinate method are all equivalent in the limit of infinite resolution or number of expansion terms. The devil, however, is in the details. These approaches give different answers at different resolutions and for different numbers of expansion terms. Here we briefly outline the major ideas of the multiflux, moment, and spherical-harmonic methods. The discrete-ordinate approach is discussed in somewhat more detail.

Multiflux Methods

If the intensity is uniform over a number of different intervals of solid angle, the RTE reduces to a series of coupled linear differential equations in terms of average radiation intensities, or fluxes. Then, as the number of different solid angles is changed, different flux methods are obtained, leading to, for example, the two-flux method, four-flux method, six-flux method, and so on. As the number of flux directions increases, the accuracy is expected to increase. For some time, these flux methods were the most common approaches to radiative heat transfer, beginning with the early work by Schuster (1905) and Schwarzschild (1906), who developed a one-dimensional, two-flux model, to be increased later to four- and six-flux models that were even extended to multidimensions.

As an example, consider the structure of the six-flux model suggested by Spalding (see Viskanta and Mengüç [1987], and references therein). Here a set of first-order differential equations are written for the six fluxes J_λ^+, J_λ^-, K_λ^+, K_λ^-, L_λ^+, L_λ^-, which refer to the total positive and negative radiation fluxes in each of the three directions. In Cartesian coordinates, they are (x, y, z) and in axisymmetric coordinates, (r, z, ϕ). These six first-order equations may be rewritten as three second-order differential equations. One of the major problems with the formulation of such models is that unless scattering is included, the fluxes for one direction are not coupled with the fluxes in other directions. As this is a source of inaccuracy in the generic formulation, these approaches are not used much now. They have, however, been extended in directions that lead to the discrete-ordinate and spherical-harmonic methods, which are described later.

Moment Approximations

In the *moment approximation*, the RTE is approximated by a hierarchy of moment equations. This is achieved by writing I as a series of products of angular and spatial functions,

$$I(x, y, z, \theta, \phi) = A_o + \sum_{n=1}^{N} \left[c_x^n A_{n,x} + c_y^n A_{n,y} + c_z^n A_{n,z} \right], \tag{13–4.14}$$

where A is a function of position (x, y, z), and c_x, c_y, and c_z, are the direction cosines defined in equation (13–2.17). As $N \to \infty$, equation (13–4.14) converges to the exact solution. This is a Taylor series expansion of I expanded in powers of direction cosines.

Generally, the lowest three moments are used, and these have specific physical significance, reminiscent of the equations of fluid dynamics. The first moment is the radiation energy density, the second moment is the radiative energy flux, and the third moment is the radiation stress tensor. For the first order in the expansion and in one spatial dimension, this approach reduces to the commonly used Milne-Eddington approximation (Milne 1930; Eddington 1959).

Spherical-Harmonic Approximations

The *spherical-harmonics approximation*, also known as the P_N *approximation* or the *differential approximation*, was first proposed by Jeans (1917) for radiation transport in stars. Although this method is tedious and cumbersome, some practitioners feel it is the most mathematically elegant. This method is very similar to the moment method, except now the moments are taken in a way that takes advantage of the orthogonality of spherical harmonics. In this approach,

$$I(x, y, z, \theta, \phi) = \sum_{n=0}^{N} \sum_{m=-n}^{n} A_{n,\lambda}^{m}(x, y, z) Y_n^m(\theta, \phi), \tag{13–4.15}$$

where

$$Y_n^m(\theta, \phi) = (-1)^{(m+|m|)/2} \left[\frac{2n+1}{4\pi} \frac{(n-|m|)!}{(n+|m|)!} \right]^{1/2} P_n^{|m|}(\cos\theta) e^{im\phi}. \tag{13–4.16}$$

Here the $\{Y_n^m\}$ are the spherical harmonics, the $\{P_n^m\}$ are associated Legendre polynomials, and M is the order. For $N = 1$, we obtain the P_1 approximation; $N = 3$, the P_3 approximation; and so on. Generally the odd order is chosen because it avoids mathematical singularities of I in directions parallel to the boundaries.

The lowest-order P_1 method reduces the RTE to a single elliptic partial-differential equation of the general form

$$\nabla^2 I_{0,\lambda} = A_\lambda \left[I_{0,\lambda} - 4\pi I_{\lambda b}(T) \right] \tag{13–4.17}$$

where $I_{0,\lambda}$ is the integral of I over Ω (defined as $\overline{I}(\lambda)$ in equation 13–2.2), $I_{\lambda,b}(T)$ was defined in equation 13–2.7, and A_λ is a coefficient that is a function of a number of properties of the medium, such as the extinction coefficient. The P_3 method uses higher-order moments of the intensity, which are integrals over the angle 4π of products of the radiation intensity and direction cosines. For axisymmetric geometries, this yields four second-order partial differential equations. For rectangular geometries, this yields six such equations.

The lowest order P_1 method has been shown to be accurate enough for optically thick media, and inaccurate as the medium becomes thinner. It also has problems when the radiation is highly anisotropic. The P_3 method produces significant improvements on this; however, it is considerably more complex and expensive to implement.

13–4.3. The Discrete-Ordinate or S_N Method

The discrete-ordinate method (DOM) was originally suggested by Chandrasekhar (1950) for astrophysical problems. It has since been developed extensively for many types of reactive flows. The method is based on differencing and iterating the RTE over the solid angle using a finite number of ordinate directions M, where M is related to the order of the approximation N through $M = 2^d N(N+2)/8$, where d is the dimension of the problem. The DOM is now commonly called the S_N *method*, although S_N originally referred to one specific form of the DOM with a simplified type of quadrature. A good general reference to the method is Carlson and Lathrop (1968).

Here we describe the DOM procedure in more detail through an example of how it works for a gray medium in Cartesian geometry. If the gas is gray, that is, properties do not depend on the wavelength of the radiation, the divergence of the radiative heat flux becomes

$$-\nabla \cdot \mathbf{q}_r = \alpha\left(\int_0^{4\pi} I(\mathbf{r}, \Omega)\, d\Omega - 4\pi I_b\right), \tag{13–4.18}$$

and the RTE may be written

$$(\Omega \cdot \nabla) I(\mathbf{r}, \Omega) = -\kappa\, I(\mathbf{r}, \Omega) + \alpha I_b(\mathbf{r}) + \frac{\sigma}{4\pi}\int_0^{4\pi} I(\mathbf{r}, \Omega')\, \Phi(\Omega' \to \Omega)\, d\Omega'. \tag{13–4.19}$$

We solve for $I(\mathbf{r}, \Omega)$ by discretizing the entire solid angle (4π steradians) using a finite number of ordinate directions and corresponding weight factors, and writing an RTE for each of the ordinate directions m, where $m = 1, \ldots M$. The integral terms are written as a quadrature involving summations over all of the ordinate directions.

Specifically, consider a Cartesian geometry (x, y, z) with direction cosines c_x, c_y, and c_z with respect to the angle Ω, as shown in Figure 13.4. Then the RTE becomes M equations,

$$c_{x,m}\frac{\partial}{\partial x} i_m + c_{y,m}\frac{\partial}{\partial y} i_m + c_{z,m}\frac{\partial}{\partial z} i_m = -\kappa\, i_m + \alpha i_b + \frac{\sigma}{4\pi}\sum_{m'} w_{m'} \Phi_{m',m} i_{m'}, \tag{13–4.20}$$

where m and m' represent the outgoing and incoming directions, respectively. The effect of scattering is incorporated through the summation over m'. The direction Ω_m can be pictured as a point on the surface of a unit sphere with which a surface area w_m is associated. The w_m represent angular Gaussian quadrature weights and satisfy the requirement that the weights sum to the surface area of the unit sphere. A total of M directions are chosen and the angular areas are measured in units of 4π, such that $\sum_{m=1}^{M} w_m = 1$.

Now consider a two-dimensional geometry, for which an arbitrary control volume is shown in Figure 13.5. Integrating equation (13–4.20) over this volume yields

$$\begin{aligned} &c_{x,m}(A_e i_{m,e} - A_w i_{m,w}) + c_{y,m}(A_n i_{m,n} - A_s i_{m,s}) \\ &\quad = -\kappa\, i_{m,p}\, \Delta V + \alpha\, i_{b,p}\, \Delta V + \frac{\sigma\, \Delta V}{4\pi}\sum_{m'} \Phi_{m',m}\, w_{m'}\, i_{m',p}, \end{aligned} \tag{13–4.21}$$

where A_n, A_s, A_e, A_w are the corresponding areas of the north, south, east, and west faces,

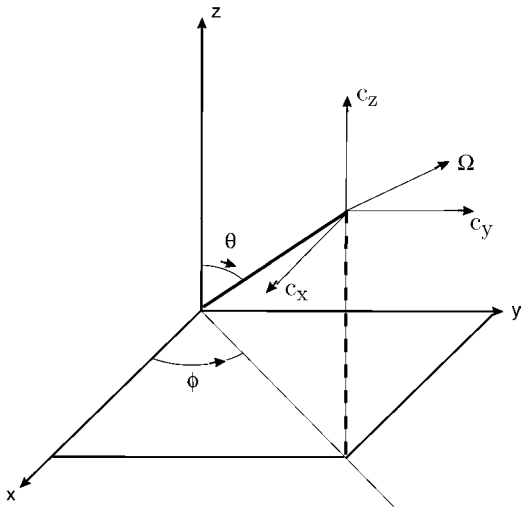

Figure 13.4. Orientation of coordinate system for radiation-transport calculation in Cartesian geometry.

respectively, of the control volume ΔV, and p is the node of interest for which we are calculating the intensity $i_{m,p}$. The terms $i_{m,n}$, $i_{m,s}$, $i_{m,e}$, and $i_{m,w}$, are the intensities for the individual mth direction at the north, south, east, and west nodes, respectively.

To solve equation (13–4.21) for $i_{m,p}$, interpolation relationships are used to relate values of intensity at unknown points to values of intensity at known points. For example, consider direction m which has positive direction cosines ($c_{x,m} > 0$, $c_{y,m} > 0$). We can eliminate the intensities at the unknown points $i_{m,n}$ and $i_{m,e}$ by expressing them in terms of intensities at known points $i_{m,s}$ and $i_{m,w}$ using

$$\begin{aligned} i_{m,p} &= \zeta\, i_{m,n} + (1-\zeta)\, i_{m,s} \quad \rightarrow \quad i_{m,n} = \frac{i_{m,p} - (1-\zeta)\, i_{m,s}}{\zeta} \\ i_{m,p} &= \zeta\, i_{m,e} + (1-\zeta)\, i_{m,w} \quad \rightarrow \quad i_{m,e} = \frac{i_{m,p} - (1-\zeta)\, i_{m,w}}{\zeta}. \end{aligned} \tag{13–4.22}$$

Then for ($c_{x,m} > 0$, $c_{y,m} > 0$), we substitute equation (13–4.22) into equation (13–4.21) and rearrange terms to give an expression for $i_{m,p}$ in terms of known variables,

$$i_{m,p} = \frac{c_{x,m}[(1-\zeta)\, A_e + \zeta A_w]\, i_{m,w} + c_{y,m}[(1-\zeta)\, A_N + \zeta A_s]\, i_{m,s} + S}{c_{x,m}\, A_e + c_{y,m}\, A_n + \kappa\, \zeta\, \Delta V}, \tag{13–4.23}$$

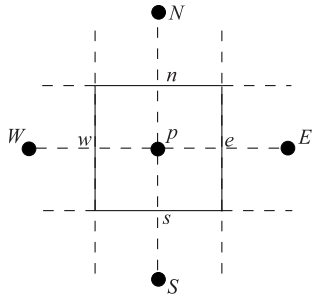

Figure 13.5. Control volume for the discrete-ordinate method of calculation in two-dimensional Cartesian coordinates.

where

$$S \equiv \zeta \Delta V \left(\alpha \, i_{b,p} + \frac{\sigma}{4\pi} \sum_{m'} w_{m'} \, \Phi_{m',m} \, i_{m',p} \right). \tag{13–4.24}$$

Once $i_{m,p}$ is calculated from equation (13–4.23), the unknown surrounding intensities, $i_{m,n}$ and $i_{m,e}$, are found from the interpolation relations equation (13–4.22). These intensities are then used as input for the calculation of intensity at the next cell in the corresponding direction, and the solution proceeds recursively through the entire grid for this direction.

Consider the S_4 approximation. Intensities of all of the points in the computational domain are calculated for each of the twelve directions considered. As outlined in the *principle of directional evaluation* (Chandrasekhar 1950), we first solve the intensity at each grid point for each direction with positive direction cosines ($c_{x,m} > 0$, $c_{y,m} > 0$), then for each direction with ($c_{x,m} < 0$, $c_{y,m} > 0$), then for each direction with ($c_{x,m} < 0$, $c_{y,m} < 0$), and finally for each direction with ($c_{x,m} > 0$, $c_{y,m} < 0$).

For ($c_{x,m} > 0$, $c_{y,m} > 0$), the calculation starts at the left bottom corner of the computational domain, and proceeds to the top right corner; known intensities are at the south and west nodes, and interpolation formulas, as in equation (13–4.22), are used to eliminate the intensity at the north and east nodes from equation (13–4.21) to calculate the intensity at the point of interest i_{mp}. For ($c_{x,m} < 0$, $c_{y,m} > 0$), the calculation proceeds from the right bottom corner to the top left corner, and known intensities are at the south and east nodes. For ($c_{x,m} < 0$, $c_{y,m} < 0$), the calculation proceeds from the top right corner to the bottom left corner, and known intensities are at the north and east nodes. Finally, for ($c_{x,m} > 0$, $c_{y,m} < 0$), the calculation proceeds from the top left corner to the bottom right corner, and known intensities are at the north and west nodes.

Central differencing ($\zeta = 0.5$) is used initially in the interpolation relationships. Even if all terms in equation (13–4.23) are positive, the interpolation relations, equation (13–4.22), could produce unphysical negative intensities. In practice this situation usually occurs in the presence of steep gradients or when the spatial resolution is inadequate. In numerical simulations, if a negative intensity is calculated, the interpolation is changed from central differencing towards upwind differencing (that is, increase ζ to somewhere between 0.5 and 1.0) until a stable solution for which there are no negative intensities is achieved.

Iterations are performed over all ordinate directions and cells until the calculated intensity at each computational cell does not change within a given tolerance depending on the individual direction considered. This progression of linear interpolations across the grid is a very numerically diffusive process, and as such, it could introduce substantial errors in a computation. Very often this is not a problem, because the radiation itself is a diffusive process and spreads out on the grid. If the radiation has a very definite direction, as it would for laser propagation, then it would be useful to align the grid for radiation along the direction of the laser beam.

There are many advantages of DOM that are particularly useful in reactive-flow codes. DOM deals with the angular dependence of I, and so takes care of directional effects. DOM works well for radiation propagation through both optically thick and thin materials. The accuracy can be varied and tested by extending the order of the solution; that is, by considering more ordinate directions. All of this, and the basic grid structure on which it

is defined, is straightforward to implement in a reactive-flow code. Finally, at significant cost, DOM can be extended to include the effects of multiple wavelength bands and lines of radiation, for which the RTE becomes a series of equations for $i_{m,p,\lambda}$. In the next section, we show an example of the use of DOM for a diffusion flame.

13–5. Using These Models for Reactive Flows

There are many reactive-flow systems for which neglecting radiation transport cannot be physically justified. If the computation is to be even qualitatively correct, the effects of the $(\nabla \cdot \mathbf{q}_r)$ term should be computed and included self-consistently in the Navier-Stokes equations. Following the general approach described in Chapter 11, $(\nabla \cdot \mathbf{q}_r)$ should be evaluated along with the change in temperature it implies, and these source terms are then combined with all of the other physical and chemical processes through an operator-splitting approach.

What is the minimal radiation model that can be used to represent radiation effects accurately enough? The answer to this question involves determining the necessary wavelength dependence that must be included, which physical approximations are appropriate, and the least expensive numerical method that is sufficiently accurate for the problem. Because computational resources are usually limited, these are not independent issues. For example, if a large number of wavelength lines and bands have to be included, treating the full directional dependence could be prohibitive. Compromises must be made, and these choices must be based on the perceived importance of different aspects of the radiation problem to the overall solution.

First consider the input absorption, emission, and scattering coefficients. The wavelength dependence of these radiation properties is taken into account by performing *multispectral calculations*; that is, by dividing the wavelength (or frequency) spectrum into a number of bands. Then it is assumed that the absorption and emission characteristics of each species remain either uniform or change smoothly according to a given functional form over the chosen spectral bands. The accuracy of this approximation increases as the bands become narrower. An exact result can only be obtained by a line-by-line calculation, which requires analysis of each discrete absorption and emission line.

In some cases, it is necessary to compute the line-by-line radiation from the RTE. This is the case in certain astrophysical computations where the details of the computed spectra are compared directly to the observational line spectra. More often, certain groupings and averaging processes are possible. For example, sometimes an important radiation line may have properties very similar to the band of radiation defined about it. In this case, the line may be grouped with the band and an appropriate averaged set of radiation properties determined. In other cases, these properties are quite different, and they must be treated differently, so that using averaged properties can mask important physical effects. In general, full spectral calculations may be used to find estimates of the absorption or emission coefficients that are to be used in the RTE calculation. This is in analogy to the problem of using a detailed chemical reaction model to derive parameters for reduced chemical mechanisms, as discussed in Chapter 5. Again, in analogy to chemical reaction kinetics, there is no unique way to do the appropriate averaging to find the coefficients, and how this is done may affect the final answer significantly. A useful assumption is that

the radiation is gray, meaning that the properties are not functions of wavelength. The details of how absorption and scattering coefficients are determined are discussed in most of the standard texts on radiation transport.

The next issue is to determine which physical assumptions about radiation are valid under the conditions considered, and use this information to determine the appropriate method for solving the RTE. In some cases, the effects of radiation transport may be decoupled, at least to a first approximation, from the reactive-flow problem and evaluated afterwards as a diagnostic. This is part of the bootstrap process for computing the spectra in the very early phase of thermonuclear supernova explosions: a dynamic model is used to predict detailed element concentrations. These concentrations are used to compute spectra, which are, in turn, compared to observations. In another limiting case, the radiation can be considered as generated locally, perhaps as a function of local temperature, and then it can be accounted for as an energy-loss term. This generally describes unconfined combustion of gases for which the amount of soot is negligible. In the most difficult cases, the interaction may be highly nonlinear and the stiff coupling must be treated accurately. This is the case for diffusion flames in which the medium varies from optically thick to thin, and there are directional effects to consider. In the worst cases, unsteady effects in the radiation transport must be considered, and there may be severe numerical issues associated with timestep control. This occurs, for example, in astrophysical systems involving core collapse in supernovae. The outer material expands so quickly that radiative energy is transferred through the material on the time scale of the expansion process itself. For most problems, the radiation equilibrates very quickly, and it is only necessary to update a quasisteady RTE at each global timestep to obtain $(\nabla \cdot \mathbf{q}_r)$.

As can be seen in the earlier parts of this chapter, some of the methods for solving the RTE are more useful than others for reactive-flow simulations. The easiest methods to use are the limiting cases, such as the transparent-gas and the diffusion approximations. The discrete-ordinate method is perhaps the most versatile of the complex methods, as it incorporates the effects of the directional dependence of radiation and the effects of a medium undergoing local transitions from thick to thin. Monte Carlo methods, while the most accurate, are the most expensive. Their best use is as stand-alone models for extremely accurate but limited computations used to find parameters for faster, less accurate models. It is prohibitive to incorporate them directly in the types of fluid-based computations described throughout this book. An example of this would be the combination of a zonal method that uses input exchange factors computed from separate Monte Carlo simulations. From the point of view of the structure of the computer program, the combination of a Monte Carlo method for radiation transport with a direct simulation Monte Carlo method (see Chapter 9) for fluid dynamics seems straightforward if expensive.

An Example: An Unsteady Ethylene Diffusion Flame

Consider a jet of ethylene gas into an air background. As the jet expands, it slows and spreads. Mixing occurs in the regions of contact between ethylene and air. The jet may be laminar if its velocity is low, or it will be turbulent at higher velocity. Given a heat source (such as a spark), the jet ignites, reacts, and burns in the mixed region along the fuel-air interface. Even when the flame is laminar, it is unsteady because of the presence of buoyancy-induced structures between the flame and the unburned gas. The major products

of combustion are water vapor, CO_2, CO, and soot. Soot is formed at about 1700 K in regions of slightly fuel-rich stoichiometry, and it is later destroyed by oxidation. When and where soot exists, the gas is optically thick.

Numerical simulation of such an unsteady jet diffusion flame is a challenge for at least two reasons: it is difficult and expensive to resolve the disparate time and space scales of the controlling processes, and there is probably not enough information available to determine sufficiently accurate input data for the level of the model. The physical and chemical processes can cover time and space scales ranging over many orders of magnitude. Even though it is usually not practical to develop a direct numerical simulation that can resolve this full range of scales, it is possible to take advantage of what is known and choose appropriate algorithms with appropriate grids.

Selected results from unsteady, axisymmetric computations (Kaplan et al. 1994a) of an unconfined ethylene jet are shown in Figures 13.6 through 13.8. The computations were performed using:

- An implicit convection algorithm (see Chapter 10) based on BIC-FCT. This was selected because of its monotone properties that provide high resolution of strong gradients while allowing the timestep to be considerably greater than the speed of sound. This involved solving an elliptic equation for the pressure or internal energy (see Chapters 9 and 10).
- Thermal-conduction, molecular-diffusion, and viscosity (see Chapters 7 and 11) coefficients that were computed from kinetic theory for a wide temperature range, and mixture coefficients that were computed from appropriate averages. Some subcycling (see Chapter 11) was used to maintain stability of the differencing algorithm because the global timestep was relatively large.
- Gravity (see Chapter 11). This physical effect was important in the low-velocity flame simulations.
- Chemical reactions and energy release (see Chapter 5), which were in the form of a phenomenological single-step reaction of the fuel and oxidizer to products. The model only tracked the species: ethylene, molecular oxygen, water, carbon dioxide, and molecular nitrogen.
- Soot production and loss, in the form of two ordinary differential equations (see Chapter 5) for the soot number density and soot volume fraction, respectively. These models contained empirically derived coefficients representing surface growth, nucleation, and coagulation. The input required is temperature and oxygen partial pressure.
- An S_n (DOM) model for radiation transport (see Section 13–4.3). Results from computations with $n = 2$, 4, and 6 were compared for this problem. The S_4 results are shown in the figures. The model assumed that a gray-gas approximation is valid; scattering was negligible compared to absorption; and only soot, carbon dioxide, and water are important for radiation.

These submodels are coupled by an operator-splitting technique for implicit fluid dynamics (see Chapter 11) that takes into account the relative stiffness of the contributing processes. The computations were performed on a fixed but nonuniform grid (see Chapter 6). The computational cells are clustered towards the lower left portion of the grid, where

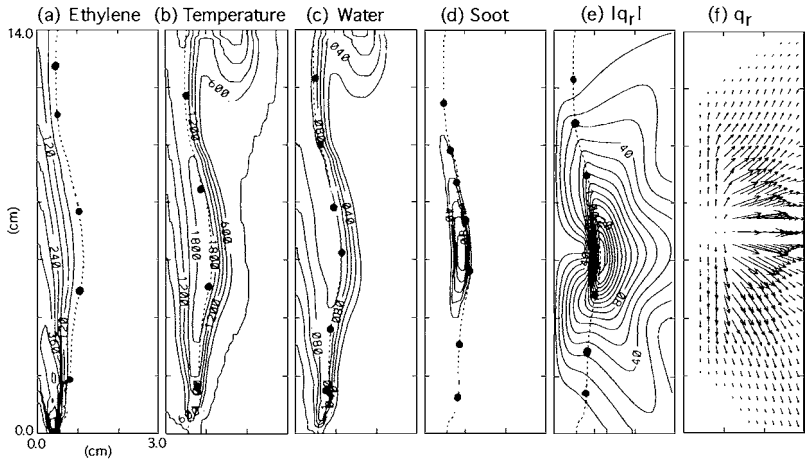

Figure 13.6. Instantaneous contours of quantities computed for 5.0 cm/s ethylene diffusion flame into background air: (a) ethylene mass fraction, (b) temperature (K), (c) water mole fraction, (d) soot volume fraction $\times 10^{-7}$, (e) magnitude of the radiative heat flux (kW/m^2), (f) radiative heat flux vectors. In (a) through (e), the location of the stoichiometric flame surface is shown by a dashed line with solid circles.

the fluid gradients are largest. In fact, only a small portion of the computational grid is shown in these figures. Various references describe a series of tests that have compared the results to theoretical limits and experimental data (e.g., Ellzey and Oran [1990], Ellzey, Laskey, and Oran [1991], Kaplan, Oran, and Baek [1994b], and Kaplan, Shaddix, and Smyth [1996]).

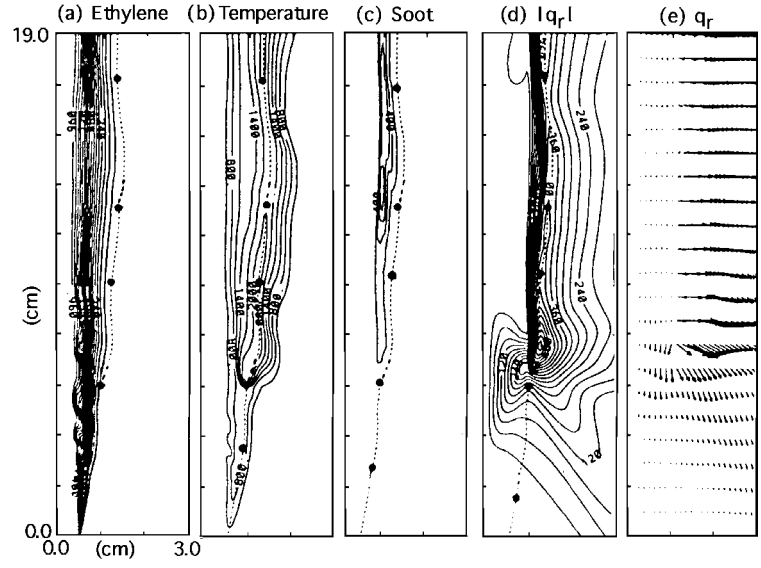

Figure 13.7. Instantaneous contours of quantities computed for 5.0 m/s ethylene jet diffusion flame into background air: (a) ethylene mole fraction, (b) temperature (K), (c) soot volume fraction $\times 10^{-8}$, (d) magnitude of the radiative heat flux $\times 10^{-1}$ (kW/m^2), (e) radiative heat flux vectors. In (a) through (d), the location of the stoichiometric flame surface is shown by a dashed line with solid circles.

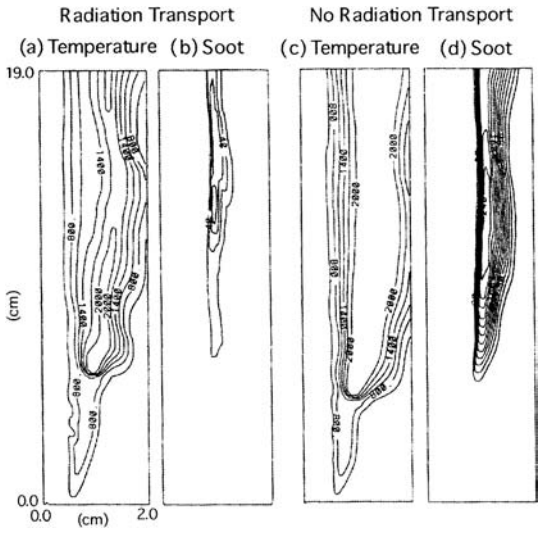

Figure 13.8. Comparison of selected quantities from computations with and without radiation transport. With radiation tranpsort: (a) temperature (K), (b) soot volume fraction $\times 10^{-7}$. Without radiation transport: (c) temperature (K), (d) soot volume fraction $\times 10^{-7}$.

First, consider the jet with 5 cm/s velocity as it enters the computational domain. Figure 13.6 shows instantaneous contours of several important physical quantities in the computation: the ethylene mass fraction, water mole fraction, soot volume fraction, the magnitude of the radiative heat flux, and the heat flux vectors. The flame surface is in the region of the 1,800–2,000 K contours. A striking feature in these figures is the large buoyancy structure that is leaving the computational domain, as new structures form below it. Soot forms in a relatively small region of the system, towards the fuel-rich region of the flame surface (inside the stoichiometric contour). This region of soot is where the absorption is highest, and the gradients of heat flux are the steepest. In addition, the S_4 model shows that the directional dependence of the heat-flux vector mirrors the curvature of the sooting region.

Figure 13.7 shows contours from a computation with a much higher jet velocity, 5 m/s. There are major qualitative differences between the flow shown here and in Figure 13.6. First, the unstable shear layer produces vortices starting near the nozzle exit, and the flame itself has lifted from the nozzle. Soot forms in a thin, narrow region starting from where the flame has lifted. The radiation shows the effects of the medium changing from optically thick to thin in the sooting region, and there are effects of the directional dependence of radiation. Figure 13.8 compares the temperature and soot volume fraction to those computed in a separate calculation in which the radiation transport is turned off. When radiation transport is not included, the flame is hotter and considerably wider, and more soot forms.

This problem is a good test of the importance of the directional properties of radiation transport. Figure 13.9 is a comparison from three computations of $|\mathbf{q}_r|$ for the same 5 m/s problem, only using progressively more angles in the DOM. The dark cores are regions of soot where the radiation is intense. The figure shows a significant difference between S_2 and S_4, and much less difference between S_4 and S_6. This is the basis of the previous statement that using S_4 was adequate for this problem. Vectors showing the directional dependence of $\mathbf{q}_r$ (not shown) indicate the change in curvature around the dark centers.

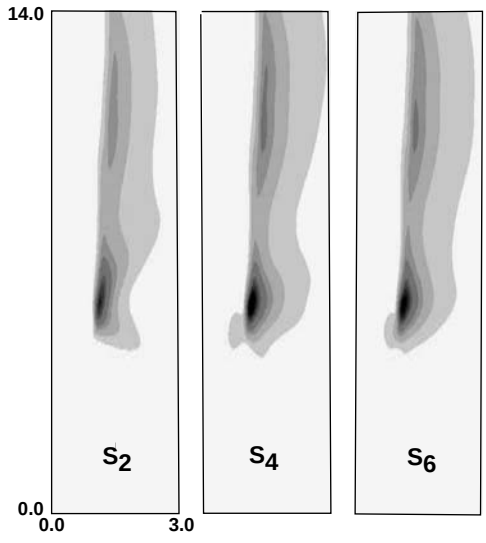

Figure 13.9. Comparison of the 5 m/s computation performed with three levels of the DOM model (S_2, S_4, and S_6) for radiation transport. The gray-scale maps show $|\mathbf{q}_r|$. The dark centers are where the soot is located and the radiation is most intense.

There are a number of factors that should be noted about these reactive-flow computations with the radiation effects included. The first and most obvious result is that there are noticeable effects of radiation transport on the prediction of important system quantities. Because of the radiation transport, the temperature decreases so the volumetric expansion decreases, and the flame shrinks in size. In the high-speed jet, there are small-scale turbulent eddies that have to be resolved to obtain the correct large-scale flow patterns. In addition, the large-scale patterns affect the production of the combustion products that emit and absorb radiation. A large number of computational cells are effectively wasted in the RTE computation if it uses the fluid-dynamics grid. In such problems, it is much more economical to use a simpler, coarser grid for the radiation transport, and then to interpolate the results onto the fluid grid. If the system had closed walls that interact with the radiation, there could be enough local heating at the walls to ignite material. This is a case in which some form of adaptive gridding would be most useful for the radiation as well as the fluid dynamics.

The diffusion flame computations described earlier include many different physical effects that are represented by very different numerical algorithms. The input data used in the physical submodels, such as the chemical reaction model, are relatively crude according to our current knowledge of ethylene combustion. Other submodels, such as those for soot formation or soot oxidation, are also rough approximations: some of the input parameters are not that well known or have been derived for another problem. Given all of these shortcomings, the results of the computations are, in some ways, quite surprising. The basic computations predict the qualitative behavior of a diffusion flame, including the effect of flame liftoff, and these results are in rather good quantitative agreement with experiments. From such studies, we gain some faith in the model for predicting flame lift heights, soot production, and radiative properties. Such a model is an excellent starting point for learning about the basic physics and chemistry of a diffusion flame, and for constructing models for practical combustion applications.

REFERENCES

Brown, G.W. 1970. Monte Carlo methods. In *Modern mathematics for the engineer*, 279–307. New York: McGraw-Hill.

Carlson, B.G., and K.D. Lathrop. 1968. Transport theory, the method of discrete ordinates. In *Computing methods in reactor physics*, eds. H. Greenspan, C. Kelber, and D. Okrent, 167–266. New York: Gordon and Breach.

Cashwell, E.D., and D.J. Everett. 1959. *A practical manual on the Monte Carlo method for random walk problems*. New York: Pergamon Press.

Chandrasekhar, S. 1950. *Radiative transfer*. London: Oxford University Press.; also, 1960, New York: Dover.

Dahlburg, J.P., M. Klapisch, J.H. Gardner, C.R. DeVore, A.J. Schmitt, and A. Bar-Shalom. 1995. Radiating plasma structures in ablating, laser-produced plasmas. *Journal of Quantitative Spectroscopy and Radiative Transfer* 54:113–121.

Eddington, A.S. 1959. *The internal constitution of the stars*. New York: Dover.

Ellzey, J.L., K.J. Laskey, and E.S. Oran. 1991. A study of confined diffusion flames. *Combustion and Flame* 84:249–264.

Ellzey, J.L., and E.S. Oran. 1990. *Effects of heat release and gravity on an unsteady diffusion flame*. In *Proceedings of the 23rd Symposium (International) on Combustion*, 1635–1640. Pittsburgh, Pa.: Combustion Institute.

Fleck, J.A. 1961. The calculation of nonlinear radiation transport by a Monte Carlo method: Statistical physics. *Methods in Computational Physics* 1:43–65.

Fureby, C., and S.I. Möller. 1995. Large eddy simulation of reacting flows applied to bluff body stabilized flames. *AIAA Journal* 33:2339–2347.

Gardner, J.H., A.J. Schmitt, J.P. Dahlburg, C.J. Pawley, S.E. Bodner, S.P. Obenschain, V. Serlin, and Y. Aglitskiy. 1998. Computational modeling of direct-drive fusion pellets and KrF-driven foil experiments. *Physics of Plasma* 5:1935–1944.

Hammersley, J.M., and D.C. Handscomb. 1964. *Monte Carlo Methods*. New York: Wiley.

Hottel, H.C., and E.S. Cohen. 1958. Radiant heat exchange in a gas-filled enclosure: Allowance for nonuniformity of gas temperatures. *AIChE Journal* 4:3–14.

Hottel, H.C., and A.F. Sarofim. 1967. *Radiative transfer*. New York, New York: McGraw-Hill.

Howell, J.R. 1968. Application of Monte Carlo to heat transfer problems. In *Advances in heat transfer*, vol. 5, eds. J.P. Harnett and T.F. Irvine. New York: Academic Press.

Jeans, J.H. 1917. The equations of radiative transfer of energy. *Monthly Notices, Royal Astronomical Society* 78:28–36.

Kalkofen, W. 1984. *Methods in radiative transfer*. New York: Cambridge University Press.

Kaplan, C.R., S.W. Baek, E.S. Oran, and J.L. Ellzey. 1994a. Dynamics of a strongly radiating unsteady ethylene jet diffusion flame. *Combustion and Flame* 96:1–21.

Kaplan, C.R., E.S. Oran, and S.W. Baek. 1994b. Stabilization mechanism of lifted jet diffusion flames. In *The Proceedings of the 25th International Symposium on Combustion*, 1183–1159. Pittsburgh, Pa.: The Combustion Institute.

Kaplan, C.R., C.R. Shaddix, and K.C. Smyth. 1996. Computations of soot production in time-varying CH_4/air diffusion flames. *Combustion and Flame* 106:392–405.

Khalil, E.E. 1982. *Modelling of furnaces and combustors*. Kent, England: Abacus Press.

Krook, M. 1955. On the solution of the equations of transfer. *Astrophysical Journal* 122:488–497.

Metropolis, N., and S. Ulam. 1949. The Monte Carlo method. *J. Am. Stat. Assoc.* 44:335–341.

Mihalas, D., and B.W. Mihalas. 1984. *Foundations of radiation hydrodynamics*. New York: Oxford University Press.

Milne, E.A. 1930. Thermodynamics of the stars. *Handbuch der Astrophysik* G. Eberhard, ed., vol. 3, part I, 65–255. Berlin: Springer.

Modest, M.M. 1993. *Radiative heat transfer*. New York: McGraw-Hill.

Noble, J.J. 1975. The zone method: Explicit matrix relations for total exchange areas. *Int. J. Heat Mass Transfer* 18:261–269.

Özişik, M.N. 1973. *Radiative transfer and interactions with conduction and convection*. New York: Wiley.

Rosseland, S. 1936. *Theoretical astrophysics: Atomic theory and the analysis of stellar atmospheres and envelopes*. Oxford, England: Clarendon Press.

Rybicki, G.B., and A.P. Lightman. 1979. *Radiative processes in astrophysics*. New York: Wiley.

Schreider, Y.A. 1964. *Method of statistical testing – Monte Carlo method*. New York: Elsevier.

Schuster, A. 1905. Radiation through a foggy atmosphere. *Astrophys. J.* 122:488–497.

Schwarzschild, K. 1906. Über das gleichgewicht der sonnenatmosphären (equilibrium of the sun's atmosphere). *Ges. Wiss.* Gottingen, *Nachr., Math-Phys. Klasse* 1:41–53.

Siegel, R., and J.R. Howell. 1968. *Thermal radiation heat transfer*, vol. 1, NASA SP-164. Washington, D.C.: National Aeronautics and Space Administration.

———. 1993. *Thermal radiation heat transfer*. New York: Hemisphere Publishing Corporation.

Viskanta, R. 1998. Overview of convection and radiation in high temperature gas flows. *International Journal of Engineering Science* 36:1677–1699.

Viskanta. R., and M.P. Mengüç. 1987. Radiation heat transfer in combustion systems. *Progress in Energy and Combustion Science* 13:97–160.

Zeldovich, Ya. B., and Yu. P. Raizer. 1966. *Physics of shock waves and high-temperature hydrodynamic phenomena*. New York: Academic.

Index

A

Absolute stability, 120, 130–134
Acceleration matching, 327
Accuracy, 77, 240
  analyzing, 77–79
  computational, 47–48
  grids and, 196–199
  limitations in, 46–48
  order and, 118
  resolution and, 196–199
  reversibility and, 109–110
  spatial, 47
  stability and, 109–110, 116–120
  See also Error; Resolution; *specific methods*
Acoustic waves, 28, 374–375
Action-at-a-distance problems, 306
Active surfaces, 354, 375–381
Adams-Bashforth methods, 124
Adams-Moulton methods, 124, 146
Adaptive mesh refinement (AMR)
  criteria for, 412
  interface and, 185–186, 376
  multidimensions and, 187–192
  overset grids, 192–196
  sliding rezone, 184
  split/merge methods, 186–187
  supersonic flow and, 293
ADI. *See* Alternating Direction Implicit methods
ADiabatic and INCompressible (ADINC) method, 321–325, 427
ADINC. *See* ADiabatic and INCompressible method
Advancing front methods, 179
Advection, 24, 94–97
Aeroacoustics, 374
ALE methods. *See* Arbitrary Lagrange-Euler methods
Aliasing errors, 273
Alternating-direction implicit (ADI) methods, 224, 398, 408, 421, 433
Amplification factor, 78, 91, 102, 224, 279
Amplitude errors, 79–80, 101–104, 196
AMR. *See* Adaptive mesh refinement
Antidiffusion, 93, 247–248, 265
Antisymmetry conditions, 359
Arbitrary Lagrange-Euler (ALE) methods, 163, 179, 331
Arnett-Truran method, 138
Arrhenius form, 25
Artificial-compressibility methods, 307
Artificial diffusion, 240
Asymptotic methods, 77, 86–88, 134–135, 220
Atmosphere, 440
Automata, 340–342
Autonomous systems, 116

B

Back substitution, 391
Backward differentiation, 132–133
Barely implicit correction (BIC) method, 250, 311, 317, 427
Beam scheme, 299
Beam-Warming method, 318
BEM. *See* Boundary-elements methods
Benchmark calculations, 56
BIC algorithm. *See* Barely implicit correction method
Binary diffusion, 229–230
Biot-Savart integral, 342
Blackbody radiation, 494
Block-implicit methods, 111, 318, 406–408
Block-structured grid, 176
Bluff-body flows, 479–480
BML model. *See* Bray-Moss-Libby model
Body-fitted grids, 174–176
Boltzmann equation, 6
Boundary conditions, 354–369
  characteristic, 374
  confined domains, 356–362
  continuitive, 363

Boundary conditions (*cont.*)
determination of, 353
edge-flux, 360–361
Euler equations and, 370
high-order algorithms, 369–375
inflow-outflow, 363–365
initial conditions and, 21
interfaces and, 353–401
Navier-Stokes flows, 370
numerical, 371
radiative, 365–366
symmetry, 356–358
unconfined domains, 362–366, 446
wave-transmission, 365–366
See also specific models
Boundary-elements methods (BEM), 382
Boundary layers, 361–362, 366–369, 443
Bray-Moss-Libby (BML) model, 470–472
Brinkman number, 30

C

Calibration tests, 196
Capillary action, 114
Cartesian grids, 180–183, 187–192
Catalytic surfaces, 362
Causality, 161, 251
Cell-centered methods, 221–224, 357
Cellular automata, 340–342
CFL. *See* Courant-Friedrichs-Lewy condition
Characteristic methods, 290, 303–305
Chebyshev method, 274, 278, 397–398
Checkerboard instabilities, 168
CHEMEQ program, 146
Chemical kinetics, 34, 82–88
CHEMKIN software, 148
Cholesky method, 398
Classical methods, for ODE's, 122–127
Clipping, 246, 251, 252
Coarsening, of grid, 183–187, 195
Coherent structures, 445
Collision integral, 229
Collisionality, 36
Collocation approximations, 270–272
Compact finite-difference methods, 239–240
Complexity
costs and, 48–50
dimensions of, 437–438
geometric, 49
gridding and, 173–183
physical system and, 48–49
programming and, 277
Compressible flows, 287–290
Compressible turbulence, 477
Compression, 94–96
Computational mesh. *See* Grids
Computational capability, 8–10
Computational singular perturbations (CSP), 151
Computer methods
accuracy and, 47–48
capabilities of, 8–10
computational cells, 46, 71
development of, 9
large-scale, 64–65
parallel, 64–70
programming in, 64–70
representation in, 44–45, 159
selecting of system, 42–44
simulation and, 11
software packages for, 145
See also specific concepts, models
Configuration-space grid, 169
Confined domains, 356–362
Conjugate gradient method, 398
Connectivity, 178, 458
Conservation, 145, 423
continuity and, 161
laws of, 75, 233
LES and, 458
physical processes and, 23–29, 37
reactive-flow, 15–23
time-dependent, 15–19
Constraints, 45–52
Contact discontinuity, 291
Continuitive methods, 363–365
Continuity equations, 161–162
conservation and, 161
continuitive forms, 363–365
convection form, 161
coupled, 104–105, 256–270
discretizing in, 163–173
integral form, 161
one-dimensional solvers, 261
physical properties, 161
positivity, 161
time reversible, 162
wave phenomena, 21, 104–108
Continuum model, 3
Contour dynamics, 306, 344–345
Convection, 20
convective flows, 159–163, 247
convective transport, 23–25, 94–104, 414–415
diffusion and, 160
finite-difference methods, 235–240
implicit methods for, 426–427
monotone methods, 246–255
reaction and, 422
simulation of, 160–161
Convergence, 116–120, 132. *See also specific methods*
Correction fluxes, 264–267
Cost, of computation, 48–50, 240
Coupling, 110–112, 405–441
advanced concepts in, 437–439
chemical differences and, 34
complexity and, 436–437
continuity equations and, 104–105, 256–270

diffusion and, 412–418
embedding and, 437–439
extensions and, 420–430
finite-element methods and, 111
fluid dynamics and, 412–418
implicit, 406–408
Lagrangian dynamics and, 428
multiple time scales, 406–411
software for, 145
species reactions and, 412–418, 424–425
stiffness and, 121, 431–436
subgrid fluctuations, 458
temporal, 436–439
thermal differences and, 34
timestep splitting, 408–409
waves and, 104–105
Courant-Friedrichs-Lewy (CFL) condition, 234
Courant number, 251, 302
Courant stability condition, 291
Covolume grids, 168
CSP. *See* Computational singular perturbations
Curl, 33
Curve fitting, 133
Cyclic reduction, 395

D

Damköhler numbers, 32, 33, 467
Defiltering, 482
Delauney triangulation, 179
Delta function, 206–210, 207
Density gradients, 463
Deterministic methods, 141
Detonation capturing, 380–381
Differential approximation, 508
Differential equations, 114–158
Diffusion
artificial, 240
binary, 229–230
boundary conditions, 360–361
coefficient of, 91, 228–231
coupling and, 412–418
defined, 204–205
divergence and, 20
effects of, 94, 417
entropy and, 94
equation for, 88–89, 205–215
fast, 220
Fickian, 224–226
molecular, 229–230
nonlinear, 215–220
physical and numerical, 93–104
process of, 93–94, 205–206
radiation and, 499–503
strong, 215–221
thermal, 230
transport, 20, 26–27, 88–94, 204–231
velocity of, 224–228
Dimensional analysis, 29
Dimensionless numbers, 29–32, 56
Directional evaluation, 511
Directional flux limiting, 265
Direction splitting, 405, 421
Direct numerical simulation (DNS), 446–448, 460–461, 468–470
Direct-simulation Monte Carlo (DSMC) methods, 7, 338
Discontinuities, at interfaces, 375–388
Discrete ordinate method, 509–512
Discrete-state cellular automata, 340
Discretization, 71–73, 159, 163–173. *See also* Grids; *specific methods*
Discriminant, 354
Dissipation, entropy and, 94
Distributed-filament vortices, 343
Distributed-memory systems, 66–67
Distribution-function methods, 452–453
Div-curl problems, 33, 289
Divergence-projection methods, 306, 309
DNS. *See* Direct numerical simulation
Documentation, 59–63
Donor-cell methods, 97, 102, 255
Double-cyclic reduction, 394–395
Douglas-Rachford scheme, 409
DSMC. *See* Direct-simulation Monte Carlo methods
Dual grid, 179
Dynamic differences, 35

E

Eddington approximation, 500–503
Eddy simulation, 453–462, 473–480
Edge-flux conditions, 360–361
Effective mass, 35
Efficiency, of models, 58–64, 226–228
Eigenvalues, of Jacobian, 129, 409
Elliptic equations, 395–399
Embedding, 439–441
Empirical models, 3
Energy, radiant, 496–497
Enthalpy, 18, 140
Entrainment, 464
Entropy, 94, 290
Equation-of-state, 21–22, 423–424
Error function, 276
Errors
algorithms and, 101–104
aliasing and, 273
amplitude and, 79, 101–104
analysis of, 46–48
convergence and, 116–120, 132
finite differences and, 79–82
grids and, 79–82, 196–198
local, 79, 196, 198
phase errors, 79, 101–104, 160
synchronization, 422–423

Errors (*cont.*)
truncation and, 48, 117
types of, 79–82
Euken factor, 228
Euler approximation, 133
Euler equations, 19, 370
Euler method, 126–130
Euler number, 30
Euler-Romberg method, 126
Eulerian grids, 162–168, 174–176
Expansion methods, 159, 507–509
collocation approximations, 270–272
finite-element methods, 275–277
Galerkin approximations, 270–272
nonlocal, 278
passive role, 464
spectral methods, 168–170, 272–278
tau approximations, 270–272
wavelet methods, 278–279
Explicit methods, 82–102, 120, 194, 221–224. *See also specific methods*
Exponential methods, 133–134
Extinction coefficient, 495
Extrapolation methods, 126–127, 136–137
Extrema, 264

F

Fast diffusion methods, 220
Fast flows, 286, 292–305
Fast-Fourier transform (FFT), 394, 447
FAST program, 503
Favre-averaged Navier-Stokes equation, 450
FCT. *See* Flux-corrected transport
Feedback, 29–34
FEMFCT method, 194
FFT. *See* Fast Fourier transform
Fick's law, 224–226
Filtering techniques, 455
Finite-difference methods, 71–82, 97. *See specific algorithms*
Finite-element methods, 111, 275–277
Finite-volume methods, 8, 75
Flame models, 311, 323, 378–379, 472–473
Flashover, 488
Fluid dynamics, 286–346
advection in, 94–96
compression in, 94–96
coupling in, 412–418
fast, 292–305
implicit, 426–428
incompressible, 305–311
Lagrangian and, 320–334, 426–428
multiphase, 35–36, 428–431
Navier-Stokes model, 426–428
slow, 292, 311–320
turbulent reactive, 443–483
See also specific types
Flux-corrected transport (FCT), 99, 234
basic concept, 247–249
causality and, 251
coupling and, 258–260
generalization of, 249–251
implicit, 314–318
limited, 267
order and, 251–254
packages for, 267–270
Reynolds number and, 277
Flux-difference splitting, 295
Flux limiting, 101, 220–221, 234, 246–256, 274
Flux synchronization, 267
Flux-vector splitting, 255, 295–299
Fourier analysis, 78, 97, 169, 253, 274. *See also* Fast Fourier transform
Fractional-step methods, 111, 309, 405. *See also* Timesteps, splitting
Free-Lagrange methods, 331–334
Free-slip conditions, 360
Frictional drag, 36
Froude number, 30
Fully threaded tree, 190–191
Fusion, 479

G

G-equation method, 383, 473
Galerkin method, 169, 270–272
Galilean invariance, 162
GAP method, 337–338
Gauss-Seidel iteration, 396
Gaussian elimination, 390
Gaussian methods, 208–209
Gear method, 408
General-connectivity grids, 179, 332
Ghost cells, 355
Gibbs errors, 79–81, 196, 197
Gibbs phenomenon, 101, 455
Glimm random choice method, 296
Global-implicit method, 111, 406–410. *See also* Block implicit methods
Global methods, 141, 143–144
Global truncation error, 117
Globally structured cartesian grids, 180–183
Godunov methods, 234, 254, 295–297, 302
Gradient methods, 185, 189, 265, 378, 398
Gravity, 28–29, 425–426
Green's function method, 143
Grids, 71, 159, 165, 195
accuracy and, 196–199
adaptive, 183–196, 293, 376, 412
body-fitted, 174–176
coarsening, 183–187
complex geometry, 173–183
configuration space, 169
connectivity of, 178
covolume of, 168

decoupling, 168
errors and, 79–82, 196–198
function on, 165–166
grid-and-particle method (GAP), 337
Lagrangian and, 164
moving, 381
overset, 176–178
particles and, 337
refining, 183–187
regular, 168
resolution and, 159–203
staggered, 168
submodels, 453–462
templates, 235
unstructured, 170–171, 178–180
Voronoi-Delauney, 168
See also Errors; *specific models, representations*
Grüneisen equation, 22
Guard cells, 355

H

Hierarchy of levels, 5–8
High-order algorithms, 234, 369–375
Homogeneity property, 297
Hottel method, 504
Hybrid methods, 145, 251–255, 405
Hydrocarbon combustion, 149
Hydrodynamics, 338–340
Hydromagnetic equations, 269

I

ICCG. *See* Incomplete Cholesky-conjugate gradient method
Ideal symmetry, 356–358
Ignition, 45, 150
Implicit methods, 82–104, 128, 353–401
cost of, 93
defined, 120
extrapolation methods, 136–137
FCT method and, 314–320
fluid dynamics and, 426–428
global, 406–408
implicit coupling and, 406–408
Lagrangian and, 323–330
leapfrog algorithm, 128
matrix inversions and, 212–213
midpoint rule, 131, 137
pressure formulations, 306–311
reversible, 108–109
Riemann solvers, 319
stiffness and, 131–133
trapezoidal method, 137
Incomplete Cholesky-conjugate gradient (ICCG) method, 398–399
Incompressible flows, 286–290, 305–311
Index space, 176
Induction parameter model (IPM), 150–151
Inertial confinement fusion, 479
Inertial range, 445
Inflow-outflow conditions, 363–365
Initial conditions, 21
Initial value problem, 115
Inspection analysis, 29
Instability, 171, 211, 463–464. *See also* Stability
Integration methods, 127–129, 145–149, 211–215
Interaction mechanisms, 29–35
Interfaces
active, 376–381
boundaries, 353–401
capturing, 375
discontinuities, 375–388
dynamics of, 294
moving grids and, 381
reactive, 464–466
tracking, 294, 376, 381
Intermittency, 406, 439–441, 446
Interpolants, 133
Intrinsic manifolds, 153–159
IPM. *See* Induction parameter model
Isotropic turbulence, 480
Iterative methods, 128–132, 226–228, 395–400

J

Jacobian matrix, 129
defined, 137
eigenvalues of, 409
iteration and, 396
JANAF tables, 140
Jet flows, 476–478, 513–514, 516–517

K

Karlovitz number, 467
Kelvin-Helmholtz instability, 460, 463
Knudsen number, 7
Kolmogorov scale, 445, 454–455, 459–461, 467, 470
Kronecker delta function, 207

L

Lagrangian methods
advantages of, 321–322
coupling and, 428
Euler methods and, 162–163
fluid dynamics and, 426–428
free, 331–334
grids, 164, 176, 332
monotonic grid, 176, 332
multidimensional, 330–331
one-dimensional implicit, 323–330
particle methods, 334–342
representations, 162–163, 170–171

Large-eddy simulation (LES), 446–447, 453–462, 468–483
Laser-driven reaction, 433–436
Lattice-gas (LG) methods, 172, 330, 340–342
Lattices, 165, 172–173. *See also* Grids
Lax-Wendroff method, 97, 100–102, 235–237, 256–258
Leapfrog methods, 97, 105–108, 127–129, 229, 237
Lennard-Jones potential, 229
LENS program, 148
LES. *See* Large-eddy simulation
Level-set approach, 383
LG methods. *See* Lattice-gas methods
Limited correction fluxes, 267
Linear multistep methods, 123–126
Liouville equation, 6
Local equations, 25
Local errors, 117, 196, 198
Local phenomena, 20, 114–158
Local processes, 82, 82–88
Local sensitivity methods, 141
Long-wavelength breakdowns, 210–211
Loss coefficient, 134
Loss rates, 18
Low-dimensional manifolds, 154–159
Lumped-parameter representation, 45

M

MAC method. *See* Marker-and-cell method
MacCormack method, 238–239
Mach cone, 274
Mach numbers, 24, 25, 30
Macroparticles, 163, 171–173
Magnetohydrodynamics (MHD), 178, 250, 304
Manifolds, dimensions of, 154–159
Marker-and-cell (MAC) method, 308–311, 337, 384
Master-slave programming, 66
Matrices, 212–213, 388–401
Mean free path, 7
Merging cells, 186–187
Mesh, 159, 165, 183–196. *See also* Grid
Message-passing interface (MPI), 66
MHD systems. *See* Magnetohydrodynamics
MILES. *See* Monotone algorithms, LES methods
Mixing, 460, 470, 476
Mixture fraction space, 468
Modular simulation, 13, 410
Molecular diffusion, 204, 229–230
Moment approximation, 507–508
Moments, method of, 276
Momentum equations, 430
Monotone algorithms, 240, 263–267
  convection and, 245–255
  flux limiting and, 265
  Lagrangian and, 176, 332–333
  LES methods, 246–256, 473–483
  nonlinear, 101
  PPM method, 97, 167, 234, 297, 477
Monte Carlo methods, 6, 498–499
Moving grids, 381
MPI. *See* Message-passing interface
Mulitiphase flows, 428–431
Multidimensional methods, 189–192, 221–224, 260–266, 330, 422
Multifluid approximation, 37
Multiflux methods, 507
Multigrid methods, 399–400
Multiphase flows, 34–68, 428–430
Multiple Fourier analysis, 393–394
Multiscale processes, 438
Multistep methods, 123–126
MUSCL method, 97

N

Navier-Stokes equations, 19, 44
  boundary conditions, 370
  Favre-averaged, 450
  fluid dynamics and, 426–428
  multiphase flow and, 35
  reactive flows, 19, 427
  turbulence-averaged, 448–451
  wave phenomena, 28
Newton-Raphson algorithm, 141
Nonequilibrium flows, 446
Nonlinearity, 29, 422
  diffusion and, 215–222
  montone methods, 101, 245, 460, 462
Nonlocal errors, 196
Nonlocal expansion, 278

O

OCI. *See* Operator compact implicit methods
ODE. *See* Ordinary differential equations
One-step methods, 122–123, 235–237
Opacity, 27, 501–502
Operator compact implicit (OCI) methods, 239
Operator splitting. *See* Timesteps, splitting
Order, 48, 118, 251–254
Ordinary differential equations (ODE), 114–158
Oscillations, 21, 104–110. *See also* Waves
Overset grids, 176–178, 192–194

P

Packages, corrected, 267–270
Padé methods, 239–240
Parallel processing, 64–70
Particle-in-cell (PIC) methods, 37, 171, 334–342, 384
Passive interface, 375
PDF. *See* Probability distribution function methods
Peaceman-Rachford scheme, 409
Peclet number, 30
Perfectly matched layers (PML), 374
Phase errors, 79–81, 101–104, 196

Phase function, 495
Phase interactions, 34–35
Phase shift, 78
Phenomenological models, 3, 50–52
Physical processes, 19–29, 205–206, 411–420
PIC. *See* Particle-in-cell methods
Piecewise-constant representation, 166–167
Piecewise-cubic splines, 169
Piecewise-parabolic method (PPM), 97, 167, 234, 297
Planck-averaged opacity, 501–502
PML. *See* Perfectly matched layers
Poisson equation, 289, 309, 344, 393–395
Positivity, 76, 161, 240, 243, 247
PPM. *See* Piecewise-parabolic method
Prandtl number, 30
Predictor-corrector methods, 124, 136, 146
Pressure formulations, 306–311
Primitive equations, 287
Probability distribution function (PDF) methods, 452–453
Programming guidelines, 60–70
Projection methods, 306, 308–310
Pseudocompressibility, 306–310
Pseudodifferential operator, 366
Pseudospectral method, 273–274

Q

QSS. *See* Quasi-steady-state methods
Quasi-steady-state (QSS) methods, 135–136
Quasiparticle methods, 335–338

R

Radiant energy flux, 496–497
Radiation boundary conditions, 365–366
Radiation transport, 27–28, 488–517
  defined, 20–21
  diffusion approximations, 499–503
  energy flux, 496–497
  equations for, 492–499
  expansion methods, 507–508
  flux methods, 496–497, 507–508
  physical basis of, 489–492
  total intensity, 493
  zonal method, 504–505
Radiative transfer equation (RTE), 492–495
Random vortex methods, 343
RANS. *See* Reynolds-averaged Navier-Stokes equations
Rational extrapolation method, 127
Rayleigh-Taylor instability, 29, 171, 463, 479
Reaction mechanisms, 114–158
Reactive interface dynamics, 464–466
Reduced reaction mechanisms, 149–154
Refining, of grids, 183–187
Relaxation time, 134
Renormalization, 422, 423
Renormalization group (RNG) models, 451, 456
Residual function, 276
Resolution, 184
  accuracy and, 196–199
  grids and, 159–203
  merging and, 186
  representations, 159–203
  Reynolds-number limitations, 279–281
Reversibility, 77, 108–110
Reynolds-averaged Navier-Stokes (RANS) equations, 447
Reynolds number, 30, 279–281, 444
Reynolds stress, 449–452
Richtmyer methods, 237–238
Riemann problem, 260, 295–297
Riemann solvers, 101, 263, 296, 319
RNG. *See* Renormalization group models
Roof function, 167, 275
Rosseland coefficient, 500
RTE. *See* Radiative transfer equation
Runge-Kutta methods, 123

S

S_N method, 509–512
Saha equation, 22
SAIM. *See* Selected asymptotic integration method
Scalar-field equation, 383
Scale evolution models, 451
Schmidt number, 30
Selected asymptotic integration method (SAIM), 146
Semi-implicit method for pressure-linked equations (SIMPLE), 319
Sensitivity analysis, 141–145
Shape function, 275
Shared-memory systems, 66
Shock wave, 79, 111, 274–294, 303–305, 377–378
SIMPLE. *See* Semi-implicit method for pressure-linked equations
Simple line interface calculation (SLIC) algorithms, 385
Single-fluid model, 36
Singular perturbations, 151–153
SLIC. *See* Simple line interface algorithms
Sliding rezone method, 184, 189
Slope limiting, 256
Slow-flow methods, 286, 292, 311–320
Smooth-particle hydrodynamics (SPH), 171, 330, 338–340
Smoothing, dissipation and, 242
Sod problem, 299
SOR. *See* Successive over-relaxation method
Sound waves, 104, 305
Species diffusion, 224–228
Species reactions, 20, 412–418
Specific heats, 30
Spectral elements, 168–170, 273–278
Spectral intensity, 492
Spectral method, 373
Spherical-harmonics approximation, 508–509

SPH method. *See* Smooth-particle hydrodynamics
Spline methods, 169, 239
Split-direction methods, 224
Split-step method, 263
Spurious mode problem, 168
Square-wave convection, 295
Square-wave test, 96–101, 249
Stability
  absolute, 120, 130–134
  accuracy and, 109–110, 116–120
  amplification factor, 244
  convergence and, 116–120, 118
  criteria for, 121
  density gradients and, 463
  diffusion and, 211
  positivity and, 244
  Rayleigh-Taylor, 171
  reversibility and, 109–110
  stiffness and, 129–131
  superstability, 137
Staggered grids, 168
Staggered leapfrog algorithm, 105–108
Staircasing effect, 181
Stiffness, 114, 120–122
  convergence and, 132
  coupling and, 121, 431–436
  defined, 46, 121
  implicit methods and, 131–132
  iteration and, 132
  solving, 129–139
  stability and, 129–131
Stochastic methods, 141, 456
Stress-equation models, 452
Strong diffusion, 215–221
Subcycling, 419–420
Subgrid models, 453–462, 469–470
Successive over-relaxation (SOR) method, 396–397
SUPERBEE method, 297, 300, 302
Supersonic flow regimes, 293
Superstability, 137
Surface phenomenologies, 361–362
Surface-tracking methods, 381–384
Symmetry, 356–359. *See also* Boundary conditions
Synchronization, 260, 267, 422

T

Tau approximations, 270–272
Taylor series, 119–124, 508, 510–511
Teepee function, 167, 275
Temperature equations, 140–141
Temporal coupling, 436–439
Tent function, 167, 275
Terracing, 251, 252
Tetrahedral grids, 180
Thermal boundary layer, 366–369
Thermal conductivity, 204, 228–229
Thermal diffusion, 230
Thermodynamics, 149
Three-step method, 311
Time-dependent methods, 15–19, 58–64, 71–73, 437–439
Time integration, 291–292
Time-reversible equations, 162
Timesteps, 46, 78, 146
  control of, 418–420
  coupling, 406–411
  multidimensions and, 260–263
  physical situations, 419
  selection, 413–414
  splitting, 111, 261, 296, 405–410, 427–432, 440
  stiffness, 432–435
  subcycling and, 419
Top-hat basis function, 168
Total-variation diminishing (TVD) methods, 97, 234, 255–256
Trade-offs, 52–53, 409–41
Transient laminar phenomena, 458
Transonic regime, 293
Transparent gas approximation, 497–498
Transport
  coefficients of, 228–231
  convective, 23–25
  diffusive, 26–27, 204–231, 204–231
  evolution models, 452
  flux-corrected, 247–249
  gradients of, 27
  radiation. *See* Radiation transport
  See also specific types
Trapezoidal method, 124, 131, 134, 137
Tree methods, 179
Tridiagonal matrices, 212–213, 391–393
Truncation error, 48, 117
Turbulence, 443–487
  boundary layers and, 443
  closure models, 447
  combustion and, 466–473
  compressibility and, 477
  flux limiting and, 256
  instability and, 463
  LES methods and, 468–470
  Navier-Stokes equations and, 448–451
  one-equation models, 451
  nonreactive, 444–453
  reactive flows and, 443–483
  Reynolds-stress models, 451–452
  subgrid models, 457–459
TVD. *See* Total-variation diminishing methods
Two-equation models, 451
Two-step method, 238

U

Unconfined systems, 362–366
Uniform boundary algorithm, 360
Units, defined, 17

Unstructured grids, 170–171, 178–180, 194–196
Upwind schemes, 101, 234, 255, 294

V

Variable Eddington model, 500
Variable spacing, 213
VCE. *See* Virtual cell embedding
Velocity potential, 23
Virial equation, 22
Virtual cell embedding (VCE), 181, 269
Viscosity, 204, 230–231
VOF method. *See* Volume-of-fluid method
Volume-of-fluid (VOF) method, 384, 386
Volume-tracking methods, 384–387
Voronoi-Delauney grids, 168
Voronoi meshes, 178
Vortex methods, 306, 342–346
Vorticity, 33

W

Wavelet methods, 170, 278–279
Waves
  boundary conditions, 365–366
  continuity equations, 21
  coupling, 104–105
  Navier-Stokes equations, 28
  oscillations and, 104–110
  properties of, 20, 28–29
Weight functions, 169, 276

X

X-space, 169

Y

YASS method, 147–148, 416

Z

Zalesak formula, 263, 265
Zero-equation models, 451
Zero-slip condition, 361
Zonal method, 504–507
Zones, 46, 71. *See also* Grids; Mesh